JOINT SIMULATION SYSTEM (JSIMS) MARITIME

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)
NP PHASE

JM-SRS-NP-0001-R0C2
3 June 1997

PREPARED BY:
Dave Bingham, Jay Caldwell, and Ralph Nebiker
JSIMS Maritime Domain Design IPT
PREPARED FOR:
Dan Bacon
JSIMS Maritime Domain Design IPT

COORDINATION/APPROVAL SHEET
FOR
JSIMS MARITIME
SOFTWARE REQUIREMENTS SPECIFICATION (SRS)
NP PHASE

DEVELOPED BY:____________________________DATE: ___________

OBJECT ANALYSTS
JSIMS Maritime Domain Design IPT
Dave Bingham, Jay Caldwell and Ralph Nebiker

APPROVED BY:____________________________DATE: ___________

IPT Leader
JSIMS Maritime Domain Design IPT
Dan Bacon

APPROVED BY:____________________________DATE: ___________

JSIMS Maritime Systems Engineer
JSIMS Maritime
Laura Knight

RECORD OF CHANGES

Version Number	Date	Description
1.0	4/24/97	Original document.
R0C1	5/1/97	Incorporate SRR Comments
R0C2	6/2/97	Correct traceability and qualification requirements

1.0 SCOPE

The scope of this effort is to specify the functionality of Modeling and Simulation (M&S) software for JSIMS Maritime Wargaming and Training, Navy Prototype (NP) phase. A framework shall be developed with which to do this and with which to produce new models and simulations. No classified software shall be developed, although data the software uses can be classified. Human System Integration (HSI) shall be accomplished through the use of a separate Work Station, housing the necessary Human Computer Interface (HCI) software and supporting utilities. The HCI software shall be hosted on TAC-4 hardware for the Geo-Tactical Display and the simulation objects software shall be hosted on Sun SPARC hardware or equivalent hardware for the simulation models. This phase of the development, NP, is to demonstrate the process for developing simulation objects and as such the scope of the software requirements are limited to a few objects. The following objects shall be implemented:

1. Ship Hull: Generic surface ship hull that shall be capable of motion.
2. FA18C: Generic FA18C aircraft without weapon or sensor systems, but capable of motion.
3. Red Aircraft. Generic Opposing forces aircraft without weapon or sensor systems, but capable of motion.
4. Identification Friend or Foe (IFF): Generic IFF system capable of interrogation which can be associated with the ship hull, and capable of responding to interrogation and being associated with FA18C and Red Aircraft objects. The IFF system shall be capable of identification of friendly tracks which properly respond to interrogation.

1.1 IDENTIFICATION

This specification produces a Framework for Wargaming and Training Simulations. It is identified by JM-SRS-NP. This requirements document describes the first version and the initial release. The basic purpose of the prototype development is to exercise and evaluate JSIMS Maritime systems engineering processes to the maximum extent practicable so as to improve those processes and ensure that they are fully understood by those who shall be responsible for executing them. Henceforward, this phase shall be referred to as JSIMS Maritime Software Segment, Navy Prototype, (JMSS NP)

1.2 SYSTEM OVERVIEW

The purpose of the JSIMS Maritime prototype development process are as follows:

1. Exercise the planned methodologies for the development of JSIMS Maritime mission space objects to the maximum extent possible. This includes integration of USMC Development Agent activities with JSIMS Maritime, interaction with the Defense Intelligence Agency (DIA) and other JSIMS Maritime Development Agents, and interaction with the JSIMS JPO and I&D contractor. (If the prototype development process interferes with resources that must be applied to perform JSIMS Maritime tasks needed to be directly applied to JSIMS development efforts, the prototype development process shall be scaled back accordingly.)

2. Test the communications methods that are planned to be used to execute JSIMS Maritime development efforts, including e-mail, teleconferences, web sites, and when available, the Enterprise Information Management System (EIMS).

3. Evaluate alternative tools which can be applied in JSIMS Maritime development efforts.

4. Exercise the JSIMS Maritime technical approach in a manner that is believed to be the most effective from the JSIMS Maritime viewpoint.

5. Complete the prototype development in about three months.

1.3 DOCUMENT OVERVIEW

This document defines and records the system-wide requirements decisions (that is, decisions about the systems behavioral design and other decisions affecting the selection of system components). The result will include all applicable items in the system-wide requirements. Requirements pertaining to interfaces and databases will be included in this SRS rather than in interface design descriptions (IRSs) and requirements pertaining to databases will also be included in the SRS rather than in database requirements descriptions (DBDDs). This document will specify all requirements and provide for demonstrating this fulfillment through qualification testing. This document will further define and record the architectural requirements of the system (identifying the components of the system, their interfaces, and a concept of execution among them) and the traceability between the system components and system requirements as expressed in the Software Development Plan (SDP). The result will include all applicable items in the architectural requirements and traceability sections. This document will also define and record the architectural requirements of the JMSS NP (identifying the software components comprising the JMSS NP JMSS NP , their interfaces, and a concept of execution among them) and the traceability between the software components and the JMSS NP requirements. The result will include all applicable items in the architectural requirements and traceability sections of the Software Design Description (SDD), design pertaining to interfaces will be included in SDDs. System requirements will be interpreted to mean the system requirements identified for this build. For purposes of this document, a software component will be construed to mean Object Class. This system is UNCLASSIFIED and will not be CLASSIFIED until the inclusion of actual data that may be classified. At such time the system will be operated in a benign environment and as such no special software features are required. There are no privacy issues with respect to this development.

2.0 REFERENCED DOCUMENTS

a Software Development Plan (SDP) for Joint Simulation System (JSIMS) Maritime Software Segment (JMSS) REVISION DATE: 31 January 1997; available from Jeff Wallace, NRaD Code D44202, San Diego, CA 92152; 619.553.6809

b .Joint Simulation System Maritime Prototype Process Model Exposition (Build NP WBS 1.2.2.2) Draft Version 1.04 April 1997; available from Ralph Nebiker, NRaD Code D44202, San Diego, CA 92152; 619.553.3971

c SOFTWARE DEVELOPMENT AND DOCUMENTATION MIL-STD-498 5 December 1994 Superseding DOD-STD-2167A 29 February 1988 DOD-STD-7935A 31 October 1988 DOD-STD-1703(NS) 12 February 1987

d Shlaer, S. and S. J. Mellor, *Object Life Cycles: Modeling the World in States*, 251 p., Yourdon Press, Englewood Cliffs, NJ, 1992.

e Leonard, G.E., L. J. Peterson and J. F. Caldwell, using the Model-View-Controller Framework as a Simulation Development Methodology, proceedings *of the Object-Oriented Simulation Conference (OOS ô97)*ó, J. W. Wallace, T. G. Beaumariage and Y. Dessouky (eds.), Phoenix, January 1997, pp. 79-84.

f LaLonde, W. R. and J. Pugh, 1991. *Inside Smalltalk Volume II.* Prentice Hall, Englewood Cliffs, N.J.

g Shlaer, S. and S. J. Mellor, The Shlaer-Mellor Method, Technical Report pf.pb.S075, Project Technology, Inc., 1996

h NP Collective System and Task List, Ship (WBS 1.2.1.1) Rev 1.0, 12 March 1997; available from Ralph Nebiker, NRaD Code D44202, San Diego, CA 92152; 619.553.3971

i NP Collective System and Task List, Aircraft (WBS 1.2.1.1) Rev 2.0, 12 March 1997; available from Ralph Nebiker, NRaD Code D44202, San Diego, CA 92152; 619.553.3971

j System/Task Description (SD/TD), Ship (WBS 1.2.1.1) Rev 1.0, 6 March 1997; available from Ralph Nebiker, NRaD Code D44202, San Diego, CA 92152; 619.553.3971

k System/Task Description (SD/TD), Aircraft (WBS 1.2.1.1) Rev 1.0, 12 March 1997; available from Ralph Nebiker, NRaD Code D44202, San Diego, CA 92152; 619.553.3971

3.0 REQUIREMENTS

This SRS for the NP Build of the JMSS is a specification of the capabilities to be provided for this Build. For purposes of understanding, the SRS constitutes the specification of the JMSS NP as described in the Object Information model (OIM). This OIM specifies the object classes to be implemented and the associations and attributes of the object classes. Computer Software Components (CSC)s are the object classes and the Computer Software Units (CSU)s are the methods described in the Action Data Flow Diagrams (ADFD)s. JMSS NP will include a family of work stations interfacing with the Simulation Engine and servers via a local area net (LAN). The software implementation framework is the Model View Controller (MVC) paradigm of SmallTalk. The models will receive input from an associated controller. Model views will be provided to interface with the Human Computer Interface (HCI)/Map Servers.

3.1 REQUIRED STATES AND MODES

The basic architecture of the JSIMS Maritime development and test environment, depicted in Figure 1, will be a workstation interfacing with the Simulation engine and servers via a local area net (LAN). The NP Phase will not employ any remote users. The Human System Integration (HSI) is accomplished by the User terminals and employ a Geographical Tactical Display as well as a textual display through separate x-windows or separate terminals. This will constitute the Human Computer Interface (HCI) for the User. No states or modes are required. As a simulation system, the system is always in a single state having a single mode.

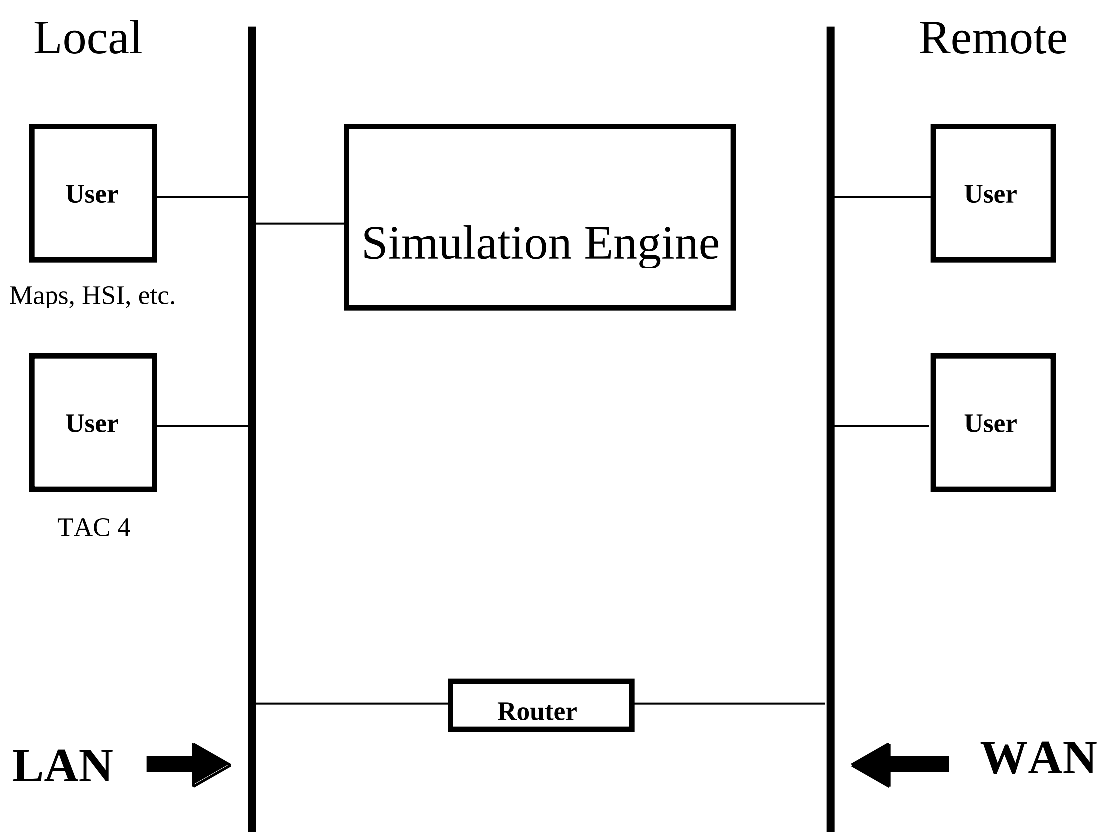


FIGURE 1. JMSS NP SYSTEMS ARCHITECTURE

The software implementation framework utilizes the Model-View-Controller (MVC) architecture of SmallTalk (See Figure 2). The models receive input from associated controllers and provide data for the views which interface with the HCI/Map Servers. External users or models will interact with the simulation for monitoring and control of the simulation, (later phases/builds). The Shlaer-Mellor Method (Refs. d, g) applied to the architecture of Figure 1 produces an integrated set of models which can be executed for verification, when using a model compiler. A model compiler will not be used in JMSS NP. Using this methodology, the design approach produces a system design through a translation of the analysis models.

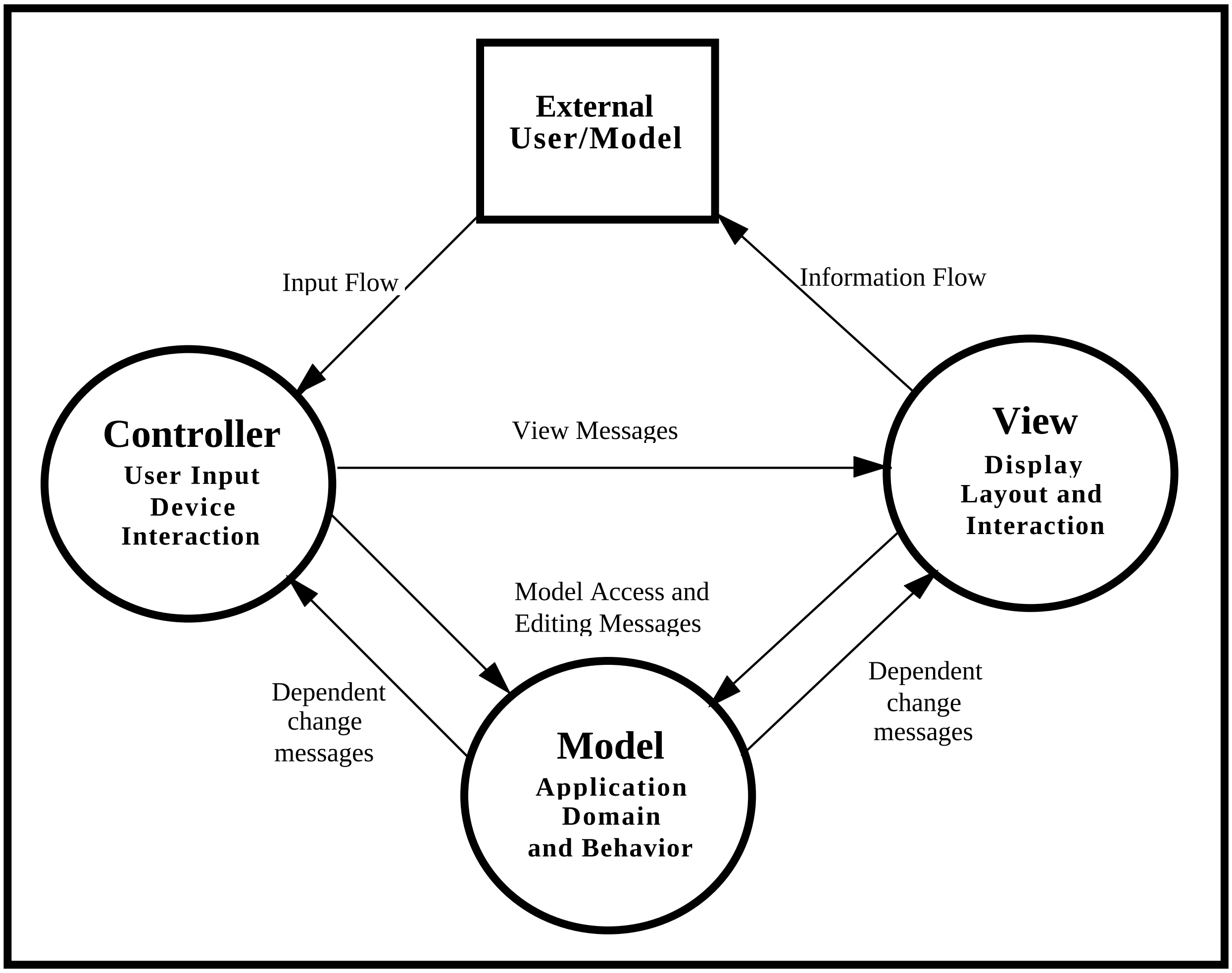


FIGURE 2. MODEL/VIEW/CONTROLLER SOFTWARE ARCHITECTURE

The Domain Chart of Figure 3 shows a partitioning of the architecture of Figure 1 into distinct domains which can be independently developed through the Shlaer-Mellor Method. Each of the ellipses represents a distinct domain. The arrows indicate client-server relationships between the domains. For instance, both the Human Computer Interface and theNP Maritime Operations Domains are clients of the Network/LAN Domain which services them and allows communications to proceed between them. The implementation domains will be implemented with government-off-the-shelf (GOTS) and commercial-off-the shelf (COTS) software. The service domains will likewise be implemented with COTS and GOTS hardware and software. The HCI will be partially GOTS furnished, partially COTS furnished, and will require some development in the C language. The NP Maritime Operations is the system to be developed.

.

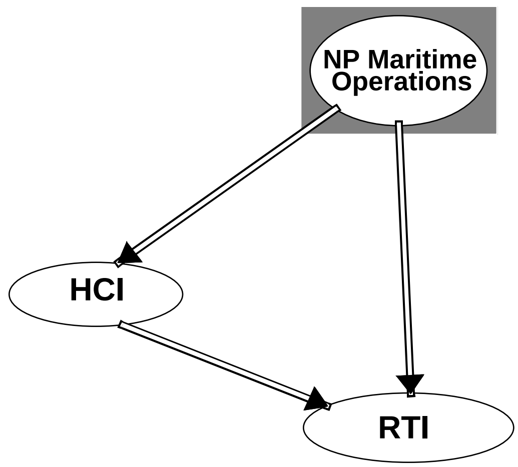


FIGURE 3. JMSS NP DOMAIN CHART

Human System Integration (HSI) is a component in the subsystem that will provide the access to the simulation model for viewing and controlling. This is shown as the Human Computer Interface (HCI) in Figure 3. There will be two views, geographical; showing the location and identification of the objects and textual; showing and providing input to the simulation model to specify the objects to be instantiated and the objects to be viewed. The output will also be provided in tabular form for the list of objects currently being tracked and viewed. The HCI is composed of four subsystems identified as the Geo-Tactical Display Module, the Status Board Module, the Order Processor Module, and the User Input Module. The Geo-Tactical Display Module manages the displaying of the entities of interest on a map display. The Status Board Module generates several formatted text displays of the status of entities of interest, including weapons remaining, position, bearing, heading, speed, etc. The HCI is considered non-deliverable software and is used for unit testing and for integration and test. As such, its design and operation will be included in an appendix to the software Desing Description (SDD).

3.2 JMSS NP CAPABILITY REQUIREMENTS

The OIM describes the scope of the JMSS NP . The Object Classes are the CSCs and the resultant CSUs are described in the ADFDs through the State Transition Diagrams (STD)s of each object class. The STDs specify the behavior of each of the object class. ADFDs are not created for every state. If the processing in the state is obvious, then a separate ADFD is not generated. If the definition of the processing can be explicitly defined in the process "bubble" of the ADFD, then a process Specification is also not required. Duplicate named events do not carry titles on both events, therefore an event that does not have a title will be found elsewhere on the STD. The external interfaces will be specified in the Object Communication Model (OCM) later in the specification.

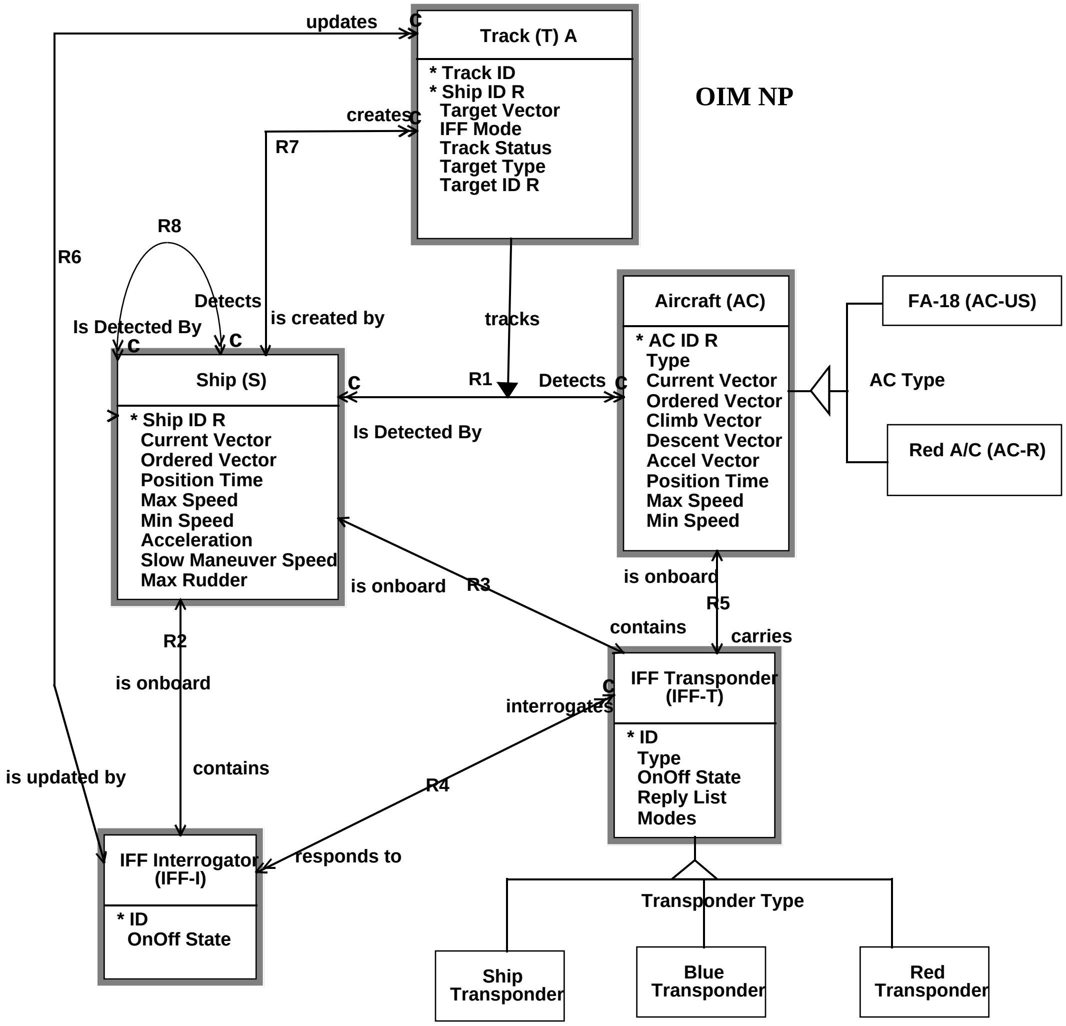


FIGURE 4. JMSS NP OBJECT INFORMATION MODEL (OIM)

3.2.1 SHIP HULL

There shall be only one ship hull object class. It shall be the only object “having a” sensor(s) capable of detection; an IFF interrogator/receiver. It shall be capable of motion, of detecting both aircraft objects, and other ships. It shall have an” IFF transponder capable of responding to IFF interrogations from other ships. See below for descriptions of the Ship Hull Object Class.

.

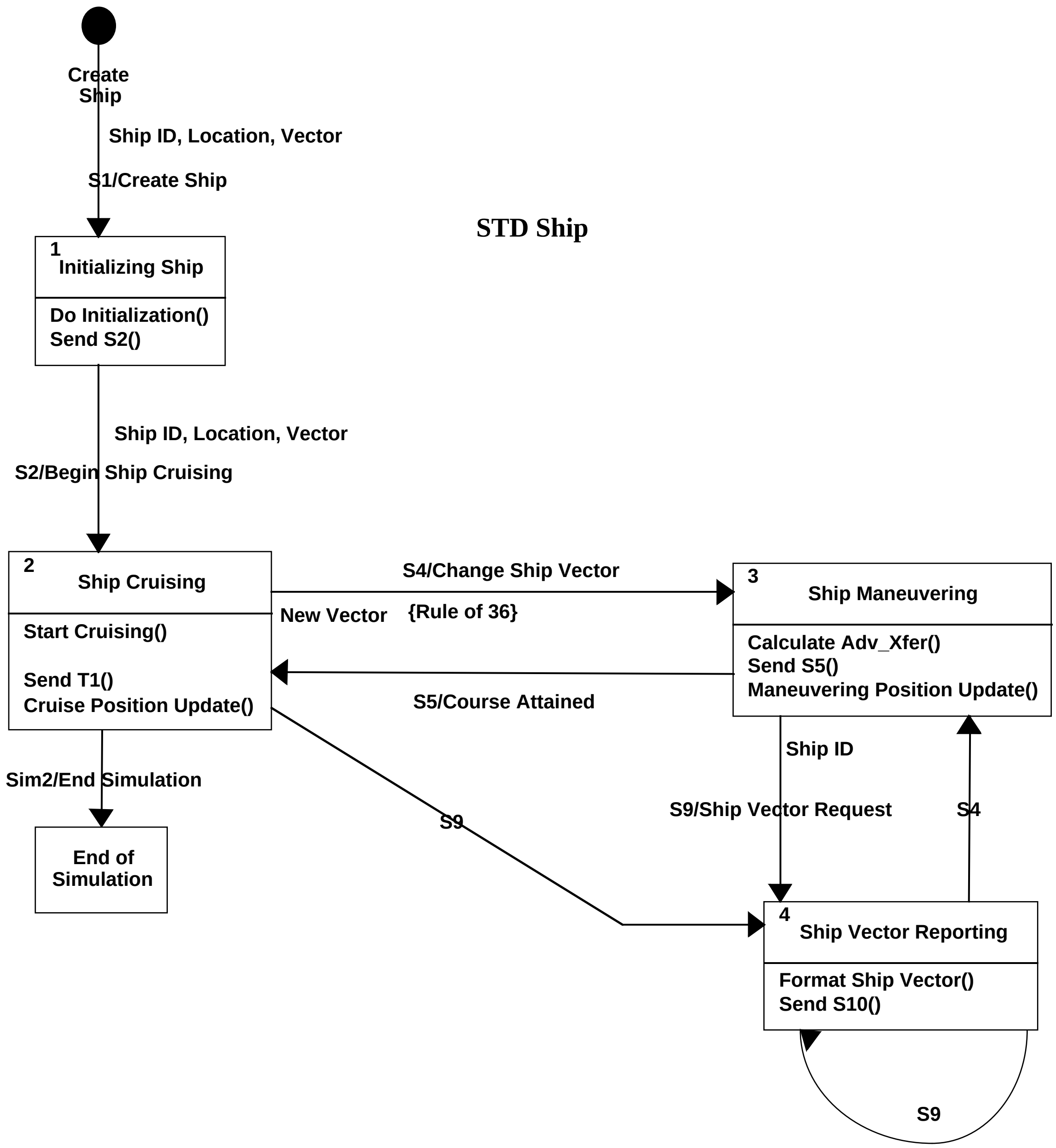


FIGURE 5. JMSS NP STD FOR THE SHIP OBJECT CLASS

:

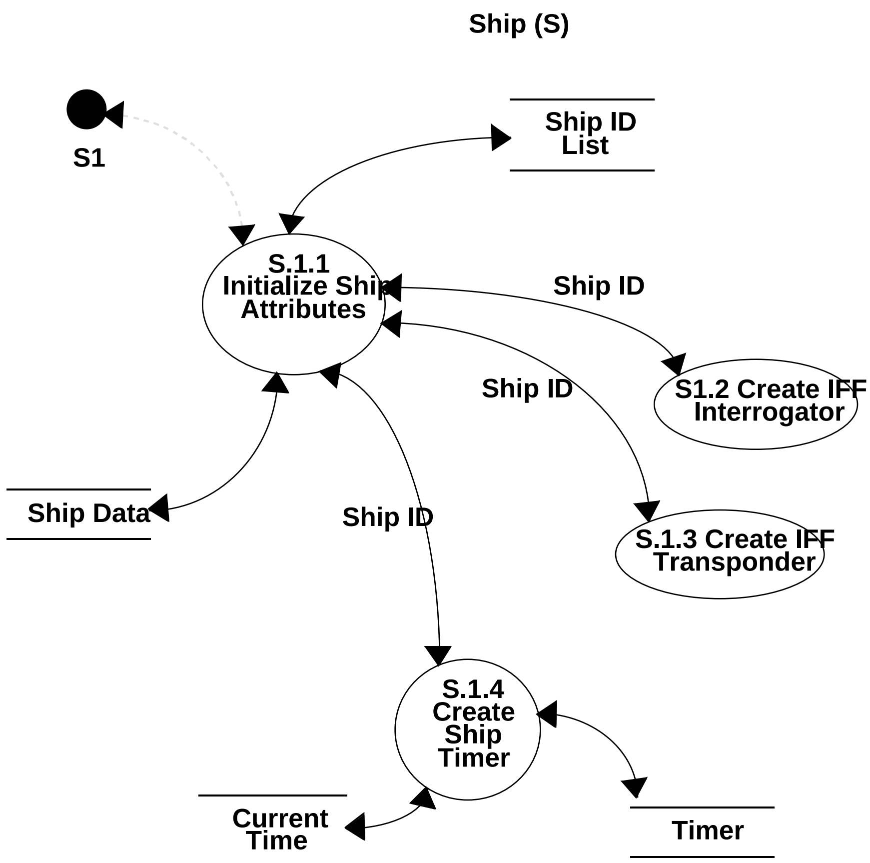


FIGURE 6. JMSS NP ADFD FOR THE SHIP OBJECT CLASS

<u>S.1.1</u>
Create_Ship (Position_Latitude, Position_Longitude)

Get Unique Ship_ID from Ship_ID_List
Type = “Ship”
Maximum_Speed = 30
Minimum_Speed = 0
Acceleration = .25
Deceleration = -.091
Slow_Maneuvering_Speed = 7
Maximum_Rudder_Angle = 35
Current_Heading = 045
Current_Speed = 15
Current_Rudder_Angle = 0
Ordered_Heading = Current_Heading
Ordered_Speed = Current_Speed
Maneuvering_State_Inidcator = “FALSE”

Time_of_Position = Current_Time
Write Data to Ship_Store
Generate S2 (Start Cruising)

End Create_Ship

<u>S.1.2</u>
Create IFF Interogator
"Send the 'create Interogator" signal"

<u>S.1.3</u>
Create IFF Transponder
"Send the 'Create IFF Transponder signal'"

<u>S.1.4</u>
Create five minute timer
Cruise_trigger
Trigger_time = current_time + five minutes

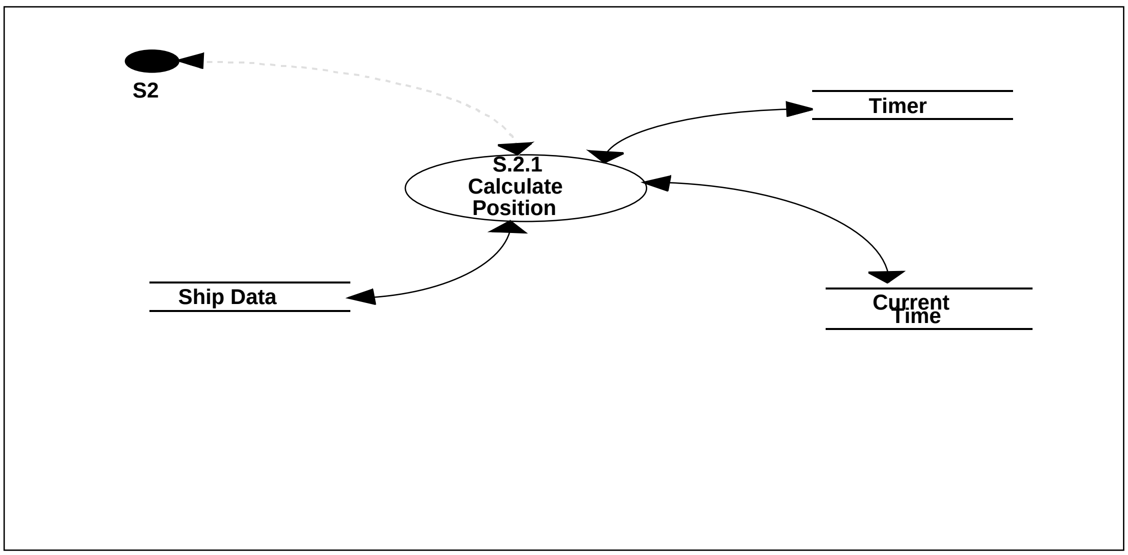


FIGURE 7. JMSS NP ADFD FOR THE SHIP CLASS

<u>S.2.1</u>
The method is invoked by timed entry every five minutes.

Method Steps in Sequence:
A. If Maneuvering_State_Indicator = True then exit.
B. Delta_Time_Min = Current_Time - Time_of_Position
C. Call Constant_Heading_Distance (Current_Heading, Current_Speed, Delta_Time_Min, North_South_Distance, East_West_Distance)
D. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)

E. Time_of_Postion = Current_Time.
F. Next_Update_Time = Current_Time plus five minutes.
G. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Position_Latitude, Position_Longitude,Time_of_Position)

Subroutine Constant_Heading_Distance (Current_Heading, Current_Speed, Delta_Minutes, North_South_Distance, East_West_Distance)

```
If Delta_Minutes < 0 then
 Delta_Minutes = 0
 North_South_Distance = 0
 East_West_Distance = 0
Else
 North_South_Distance = Delta_Minutes * Current_Speed * Cos(Current_Heading)/60.
 East_West_Distance = Delta_Minutes * Current_Speed * Sin(Current_Heading)/60.
Endif
End Constant_Heading_Distance
 Subroutine Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude,
Position_Longitude)
Convert North_South_Distance -> Latitude_Traveled.
Convert East_West_Distance -> Longitude_Traveled
Position_Latitude = Position_Latitude + Latitude_Traveled.
Position_Longitude = Position_Longitude + Longitude_Traveled.
End Convert_Distances
```

Note: Heading is based on true north; i.e. no variation. Navigation is by Rhumb line. Navigational conversions and positional updates shall be based on the map display functionality chosen by the SWEC IPT to support Build NP and the current position.

Maximum_Speed (in knots to the nearest knot; e.g. 30; HLA Real)
Minimum_Speed (in knots to the nearest knot; e.g. 30; HLA Real)
Current_Heading (in degrees to nearest degree; e.g. 243; HLA Real)
Ordered_Heading (in degrees to nearest degree; e.g. 283; HLA Real)
Current_Speed (in knots to the nearest knot; e.g. 15; HLA Real)
Ordered_Speed (in knots to the nearest knot; e.g. 25; HLA Real)
Position_Latitude (degrees to nearest second, North or South; e.g. 15-21-34N; HLA Real)
Position_Longitude (degrees to nearest second, East or West; e.g. 135-46-22W; HLA Real)
Time_of_Position (time to nearest minute; e.g. 1634; HLA Real)

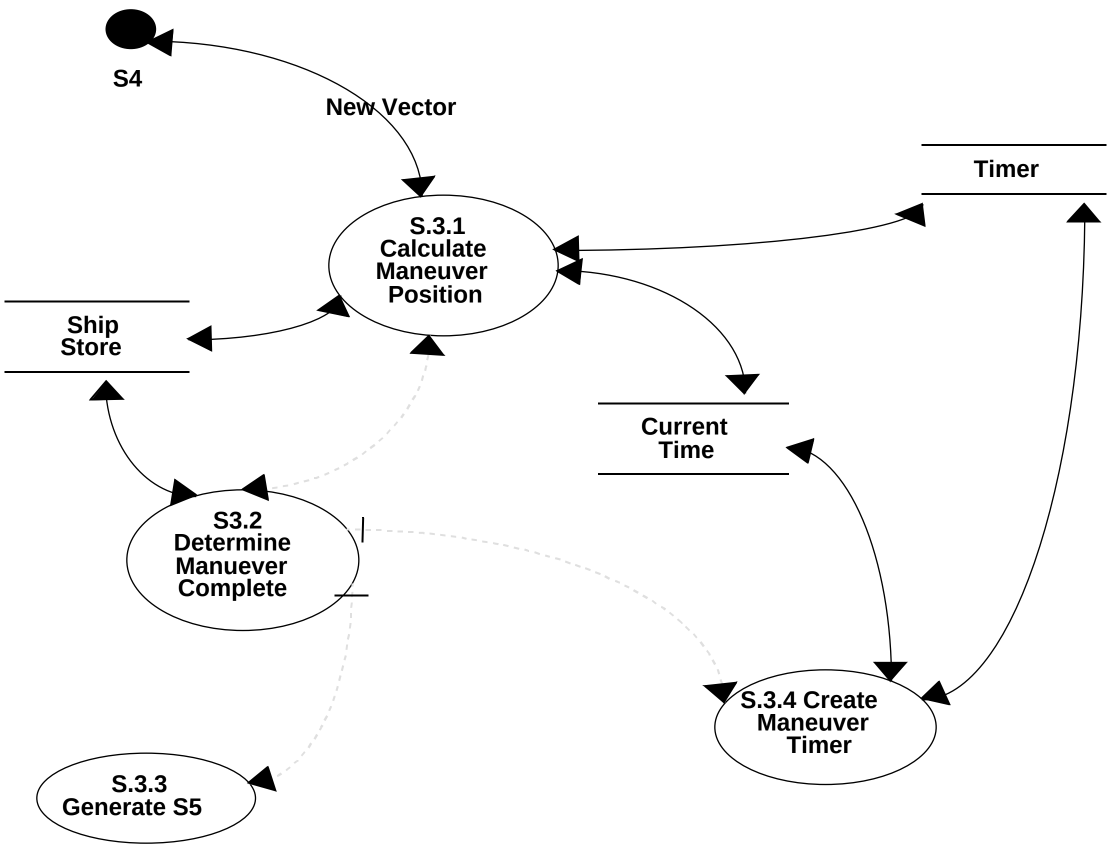


FIGURE 8. JMSS NP ADFD SHIP CLASS

<u>S.3.1</u>

This method is invoked by timed entry every minute or when the operator orders a new ship's heading, a new ship's speed, or a new rudder angle.

Timed Entry (Method Steps in Sequence):
A. If Maneuvering_State_Indicator = False then exit.
B. Call Change_Speed (Ordered_Speed, Current_Speed, Average_Speed)
C. Call Change_Heading (Current_Rudder_Angle, Ordered_Heading, Current_Heading, Average_Speed, North_South_Distance, East_West_Distance)
D. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)
E. Time_of_Position = Current_Time.
F. Next_Update_Time = Current_Time plus one minute.
G. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Position_Latitude, Position_Longitude, Time_of_Position)

New Current_Rudder_Angle Entry (Method Steps in Sequence):
Note: The procedure here is to compute current position based on the “old” ordered heading, speed, rudder angle and previous position time, and then “GET” from the operator’s command the “new” Ordered_Rudder_Angle.

A. Maneuvering_State_Indicator = True.
B. Call Change_Speed (Ordered_Speed, Current_Speed, Average_Speed)
C. Call Change_Heading (Current_Rudder_Angle, Ordered_Heading, Current_Heading, Average_Speed, North_South_Distance, East_West_Distance)
D. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)
E. GET Ordered_Rudder_Angle[Remark: Rudder angle]
F. Current_Rudder_Angle = Ordered_Rudder_Angle[changes are ‘instantaneous”]
G. If Current_Rudder_Angle < 0 then Current_Rudder_Angle = 0
H. If Current_Rudder_Angle > Maximum_Rudder_Angle then
Current_Rudder_Angle = Maximum_Rudder_Angle
Endif
I. Current_Rudder_Angle = 5 * INT(Current_Rudder_Angle / 5)
J. If Ordered_Direction_of_Turn <> Current_Direction_of_Turn then
Degrees_of_Turn_Completed = 0[Remark: New direction = new turn.]
Ordered_Heading = Current_Heading
Current_Direction_of_Turn = Ordered_Direction_of_Turn
[Remark: Ordered_Direction_of_Turn is part of ordered rudder command; L/R]
Endif
K. Time_of_Position = Current_Time.
L. Next_Update_Time = Current_Time plus one minute.
M. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Position_Latitude, Position_Longitude, Time_of_Position)

New Ordered_Heading Entry (Method Steps in Sequence):
Note: The procedure here is to compute current position based on the “old” ordered heading, speed, rudder angle and previous position time, and then “GET” from the operator’s command the “new” Ordered_Heading.

A. Maneuvering_State_Indicator = True
B. Call Change_Speed (Ordered_Speed, Current_Speed, Average_Speed)
C. Call Change_Heading (Current_Rudder_Angle, Ordered_Heading, Current_Heading, Average_Speed, North_South_Distance, East_West_Distance)
D. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)
E. GET new Ordered_Heading
F. Call Direction_of_Turn (Ordered_Heading, Current_Heading, Turn_Direction, Degrees_of_Turn)

G. If (Turn_Direction <> Current_Direction_of_Turn) OR Current_Rudder_Angle = 0 then[i.e. New Turn]
Degrees_of_Turn_Completed = 0
Current_Direction_of_Turn = Turn_Direction
Total_Degrees_of_Turn = Degrees_of_Turn
Endif
G. If Turn_Direction = Current_Direction_of_Turn then[Continue Turn]
If Degrees_of_Turn_Completed > 90 then [Start steady turn at 90 to]
Degrees_of_Turn_Completed = 90[keep calculations within]
Endif [table range of 270 degrees.]
Total_Degrees_of_Turn = Degrees_of_Turn + Degrees_of_Turn_Completed
Endif
H. Current_Rudder_Angle = 36 – Current_Speed[Remark:Rule of 36]
I. Current_Rudder_Angle = 5 * INT(Current_Rudder_Angle / 5)
J. Time_of_Position = Current_Time.
K. Next_Update_Time = Current_Time plus one minute
L. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Position_Latitude, Position_Longitude, Time_of_Position)

New Ordered_Speed Entry (Method Steps in Sequence):
Note: The procedure here is to compute current position based on the “old” ordered heading, speed, rudder angle and previous position time, and then “GET” from the operator’s command the “new” Ordered_Speed.

A. Maneuvering_State_Indicator = True.
B. Call Change_Speed (Ordered_Speed, Current_Speed, Average_Speed)
C. Call Change_Heading (Current_Rudder_Angle, Ordered_Heading, Current_Heading, Average_Speed, North_South_Distance, East_West_Distance)
D. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)
E. If Ordered_Speed > Maximum_Speed then Ordered_Speed = Maximum_Speed
F. If Ordered_Speed < Minimum_Speed then Ordered_Speed = Minimum_Speed
G. Time_of_Position = Current_Time
H. Next_Update_Time = Current_Time plus one minute
I. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Position_Latitude, Position_Longitude, Time_of_Position)

Subroutine Change_Heading (Current_Rudder_Angle, Ordered_Heading, Current_Heading, Average_Speed, North_South_Distance, East_West_Distance)

[Remark: Delta_Time_Min and Delta_Time_Sec are obtained in Change_Speed subroutine and available as global variables.]

```
If Current_Speed < Slow_Manuevering_Speed then
        [Remark: Kinematics is point to point; no advance and transfer.]
Call Constant_Heading_Distance (Current_Heading, Current_Speed,
Delta_Time_Min, North_South_Distance, East_West_Distance)
        If [ Current_Direction_of_Turn <> 0 AND
        ABS (Ordered_Heading – Current_Heading) < (15 * Delta_Time_Min) ] then
                Current_Heading = Ordered_Heading[Remark: Turn Complete]
                Current_Rudder_Angle = 0
                Current_Direction_of_Turn = 0
        Elseif [ Current_Direction_of_Turn <> 0 AND
        Ordered_Heading = Current_Heading ] then
                Current_Heading = 15 * Current_Direction_of_Turn * Delta_Time_Min
                + Current_Heading
                Ordered_Heading = Current_Heading[Remark: Constant Turn]
        Elseif [ Current_Direction_of_Turn <> 0 AND
        ABS (Ordered_Heading – Current_Heading) > (15 * Delta_Time_Min) ] then
                Current_Heading = 15 * Current_Direction_of_Turn * Delta_Time_Min
                                        + Current_Heading
        Endif                                           [Remark: Continue Turn]
EXIT SUBROUTINE
Endif           [Remark: End Slow_Speed_Maneuvering]

If Current_Rudder_Angle = 0 then
        Ordered_Heading = Current_Heading
        Current_Direction_of_Turn = 0
Call Constant_Heading_Distance (Current_Heading, Current_Speed,
Delta_Time_Min, North_South_Distance, East_West_Distance)

Elseif (Current_Heading <> Ordered_Heading) AND (Current_Rudder_Angle <> 0) then

GET Advance_&_Transfer_Table (Current_Rudder_Angle)
        [Note: Refer to Reference b for Table References.
Table Entries are every 5 degrees.]

If Degrees_of_Turn_Completed = 0 then
Advance_Completed = 0
Transfer_Completed = 0
Distance_Completed = 0
Endif

Row_Entry_Index_1 = INT  (Degrees_of_Turn_Completed / 5)
Row_Entry_Index_2 = Row_Entry_Index_1
Row_Entry_Index_3 = FIX (Total_Degrees_of_Turn / 5) + 1
If Row_Entry_Index_3 <= Row_Entry_Index_2 then
        Row_Entry_Index_3 = Row_Entry_Index_2 + 1
Seconds_Required = 0
```

```
                    Row_Entry_Index_2 = Row_Entry_Index_2 + 1
          Distance = Table_Distance_From_Start_of_Turn (Row_Entry_Index_2) –
                              Distance_Completed
                    Seconds_Required_Old = Seconds_Required
          Seconds_Required = (Distance * 3600) / (2027 * Average_Speed)
Until  [ (Seconds_Required >= Delta_Time_Sec) OR
(Row_Entry_Index_2 >= Row_Entry_Index_3) ]

Current_Heading_PM_90 = Current_Heading + 90 * Current_Direction_of_Turn
Current_Heading_PM_90 = Current_Heading_PM_90 Modulus 360

Degrees_of_This_Turn = (Row_Entry_Index_2*5) - Degrees_of_Turn_Completed
Degrees_of_Turn_Completed = Row_Entry_Index_2 * 5

If Row_Entry_Index_2 >= Row_Entry_Index_3 then
          [Remark: Turn is complete.]
          Advance_1 = Table_Advance(Row_Entry_Index_2) – Advance_Completed
Transfer_1 = Table_Transfer(Row_Entry_Index_2) - Transfer_Completed
North_South_AT = Advance_1 * Cos(Current_Heading) + Transfer_1 *
Cos(Current_Heading_PM_90)
East_West_AT = Advance_1 * Sin(Current_Heading) + Transfer_1 *
Sin(Current_Heading_PM_90)

                    Current_Heading = Ordered_Heading
                    Current_Rudder_Angle = 0
                    Current_Direction_of_Turn = 0
Else
          Fraction = (Delta_Time_Sec – Seconds_Required_Old) /
(Seconds_Required – Seconds_Required_Old)

          Advance_1 = Table_Advance (Row_Entry_Index_2 - 1)
Advance_2 = Table_Advance (Row_Entry_Index_2)
Advance_3 = Advance_1 + (Advance_2 – Advance_1) * Fraction
Advance_Completed = Advance_3 - Advance_Completed

Transfer_1 = Table_Transfer (Row_Entry_Index_2 - 1)
Transfer_2 = Table_Transfer (Row_Entry_Index_2)
Transfer_3 = Transfer_1 + (Transfer_2 – Transfer_1) * Fraction
Transfer_Completed = Transfer_3 - Transfer_Completed

Distance_1 = Table_Distance_From_Start_of_Turn (Row_Entry_Index_2 - 1)
Distance_2 = Table_Distance_From_Start_of_Turn (Row_Entry_Index_2)
Distance_3 = Distance_1 + (Distance_2 – Distance_1) * Fraction
Distance_Completed = Distance_3 - Distance_Completed

          North_South_AT = Advance_Completed * Cos(Current_Heading) +
```

Transfer_Completed * Cos(Current_Heading_PM_90)
East_West_AT = Advance_Completed * Sin(Current_Heading) +
Transfer_Completed * Sin(Current_Heading_PM_90)

Current_Heading = Current_Heading + Degrees_of_This_Turn *
Current_Direction_of_Turn
Current_Heading = Current_Heading Modulus 360
Endif

Delta_Time = (Delta_Time_Sec – Seconds_Required) / 60
Call Constant_Heading_Distance (Current_Heading, Current_Speed,
Delta_Time, North_South_Distance, East_West_Distance)
North_South_Distance = North_South_Distance + North_South_AT
East_West_Distance = East_West_Distance + East_West_AT

Elseif (Ordered_Heading = Current_Heading) AND (Current_Rudder_Angle <> 0) then
[Remark: This is a constant turn, no final heading.]

GET Advance_&_Transfer_Table (Current_Rudder_Angle)
[Note: Refer to Reference b for Table References.
Table Entries are every 5 degrees.]

If Degrees_of_Turn_Completed > 90 then Degrees_of_Turn_Completed = 90
[Remark: Ship is in a steady turn; calculate from start of steady turn circle.]

If Degrees_of_Turn_Completed = 0 then
Advance_Completed = 0
Transfer_Completed = 0
Distance_Completed = 0
Endif

Row_Entry_Index_1 = INT (Degrees_of_Turn_Completed / 5)
Row_Entry_Index_2 = Row_Entry_Index_1
Seconds_Required = 0

Do
Row_Entry_Index_2 = Row_Entry_Index_2 + 1
Distance = Table_Distance_From_Start_of_Turn (Row_Entry_Index_2) –
Distance_Completed
Seconds_Required_Old = Seconds_Required
Seconds_Required = (Distance * 3600) / (2027 * Average_Speed)
Until [(Seconds_Required >= Delta_Time_Sec)

Fraction = (Delta_Time_Sec – Seconds_Required_Old) / (Seconds_Required –
Seconds_Required_Old)
Advance_1 = Table_Advance (Row_Entry_Index_2 - 1)

```
Advance_2 = Table_Advance (Row_Entry_Index_2)
Advance_3 = Advance_1 + (Advance_2 – Advance_1) * Fraction
Advance_Completed = Advance_3 - Advance_Completed

Transfer_1 = Table_Transfer (Row_Entry_Index_2 - 1)
Transfer_2 = Table_Transfer (Row_Entry_Index_2)
Transfer_3 = Transfer_1 + (Transfer_2 – Transfer_1) * Fraction
Transfer_Completed = Transfer_3 - Transfer_Completed

Distance_1 = Table_Distance_From_Start_of_Turn (Row_Entry_Index_2 - 1)
Distance_2 = Table_Distance_From_Start_of_Turn (Row_Entry_Index_2)
Distance_3 = Distance_1 + (Distance_2 – Distance_1) * Fraction
Distance_Completed = Distance_3 - Distance_Completed

Current_Heading_PM_90 = Current_Heading + 90 * Current_Direction_of_Turn
Current_Heading_PM_90 = Current_Heading_PM_90 Modulus 360
North_South_AT = Advance_Completed * Cos(Current_Heading) +
Transfer_Completed * Cos(Current_Heading_PM_90)
East_West_AT = Advance_Completed * Sin(Current_Heading) +
Transfer_Completed * Sin(Current_Heading_PM_90)
North_South_Distance = North_South_AT
East_West_Distance = East_West_AT

Degrees_of_This_Turn = (Row_Entry_Index_2*5) - Degrees_of_Turn_Completed
Degrees_of_Turn_Completed = Row_Entry_Index_2*5
        Current_Heading = Current_Heading + Degrees_of_This_Turn *
Current_Direction_of_Turn
        Current_Heading = Current_Heading Modulus 360
Endif
End Change_Heading


Subroutine Change_Speed (Ordered_Speed, Current_Speed, Average_Speed)

Delta_Time_Min = Current_Time – Time_of_Position  [In MIN !!!]
Delta_Time_Sec = Current_Time – Time_of_Position  [In SEC !!!]

If Ordered_Speed = Current_Speed then
Average_Speed = Current_Speed
Elseif Ordered_Speed > Current_Speed then
Seconds_Required = (Ordered_Speed – Current_Speed) / Acceleration
If Seconds_Required > = Delta_Time_Sec then
If (Seconds_Required - Delta_Time_Sec) < 5 then[Remark: 5 sec =]
New_Speed = Ordered_Speed[85 ft at 30 kts; < 1 sec arc;]
                Else                                            [acceleration completed]
        New_Speed = Current_Speed + Acceleration * Delta_Time_Sec
```

```
Endif
Average_Speed = .5 * (New_Speed + Current_Speed)
Current_Speed = New_Speed
Elseif Seconds_Required < Delta_Time_Sec then
If (Delta_Time_Sec - Seconds_Required) < 5 then
New_Speed = Ordered_Speed
                    Else
          New_Speed = Current_Speed + Acceleration * Seconds_Required
Endif
Average_Speed = .5 * (New_Speed + Current_Speed)
Average_Speed = (Average_Speed – New_Speed)*Seconds_Required /
Delta_Time_Sec
                    Average_Speed = New_Speed + Average_Speed
Current_Speed = New_Speed
          Endif
Elseif Ordered_Speed < Current_Speed then
Seconds_Required = (Current_Speed – Ordered_Speed) / Deceleration
If Seconds_Required > = Delta_Time_Sec then
If (Seconds_Required - Delta_Time_Sec) < 5 then
New_Speed = Ordered_Speed
                     Elseif
New_Speed = Current_Speed + Deceleration * Delta_Time_Sec
Endif
Average_Speed = .5 * (New_Speed + Current_Speed)
Current_Speed = New_Speed
Elseif Seconds_Required < Delta_Time_Sec then
If (Delta_Time_Sec - Seconds_Required) < 5 then
New_Speed = Ordered_Speed
                    Else
          New_Speed = Current_Speed + Deceleration * Seconds_Required
Endif
Average_Speed = .5 * (New_Speed + Current_Speed)
          Average_Speed = (Average_Speed – New_Speed)*Seconds_Required /
Delta_Time_Sec
                    Average_Speed = New_Speed + Average_Speed
Current_Speed = New_Speed
Endif
Endif
End Change_Speed
```

Subroutine Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)

Note: Heading is based on true north; i.e. no variation. Navigation is by Rhumb line. Navigational conversions and positional updates shall be based on the map display functionality chosen by the SWEC IPT to support Build NP and the current position.

```
Convert North_South_Distance -> Latitude_Traveled.
Convert East_West_Distance -> Longitude_Traveled
Position_Latitude = Position_Latitude + Latitude_Traveled.
Position_Longitude = Position_Longitude + Longitude_Traveled.
End Convert_Distances

Subroutine Direction_of_Turn (Ordered_Heading, Current_Heading,
Turn_Direction, Degrees_of_Turn)

Turn_Direction = 1

If Ordered_Heading = Current_Heading then Turn_Direction = 0
        [Remark: Right = 1, Left = -1, No turning = 0]

Degrees_of_Turn = Ordered_Heading – Current_Heading

If Degrees_of_Turn > 180 degrees then
Turn_Direction = -1
Degrees_of_Turn = 360 - Degrees_of_Turn
Endif
If Current_Heading > Ordered_Heading then
        Degrees_of_Turn = Current_Heading – Ordered_Heading
        If Degrees_of_Turn < 180 degrees then
Turn_Direction = -1
        Else
                Degrees_of_Turn = 360 - Degrees_of_Turn
        Endif
Endif
End Direction_of_Turn
```

<u>S.3.2</u>

```
If Current_Heading = Ordered_Heading AND Current_Rudder_Angle = 0 AND
Current_Speed = Ordered_Speed  then
                Next_Update_Time = Next_Update_Time + 4 minutes
                Maneuvering_State_Indicator = False
Endif
```

<u>S.3.3</u>

Generate and send the event S5

<u>S.3.4</u>

Set Ship Timer
Get current time
add 5 minutes
Set periodic ship timer

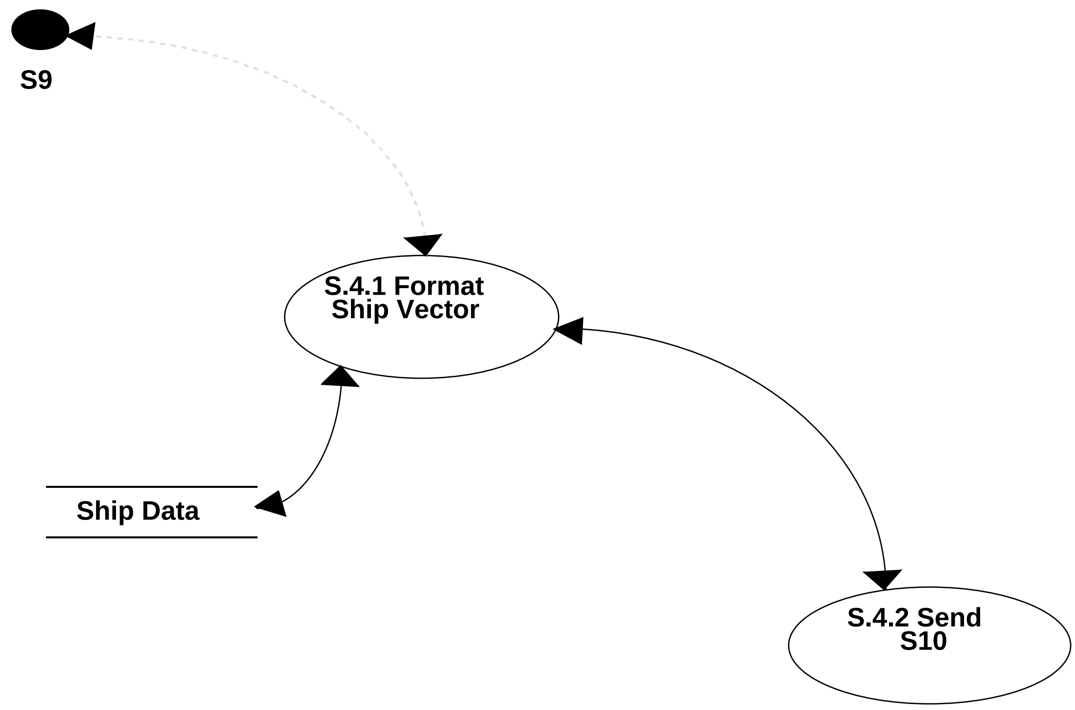


FIGURE 9. JMSS NP ADFD SHIP CLASS

S.4.1
Receive event to output ship vector
Get ship data from ship data store
Format ship data

S.4.2
Receive formatted ship data
Generate event S10

3.2.2 AIRCRAFT OBJECT CLASS

There shall be only one FA18C object. It is instantiated from the Aircraft object class. It shall "have an" IFF transponder capable of responding to IFF interrogations from ships.

.

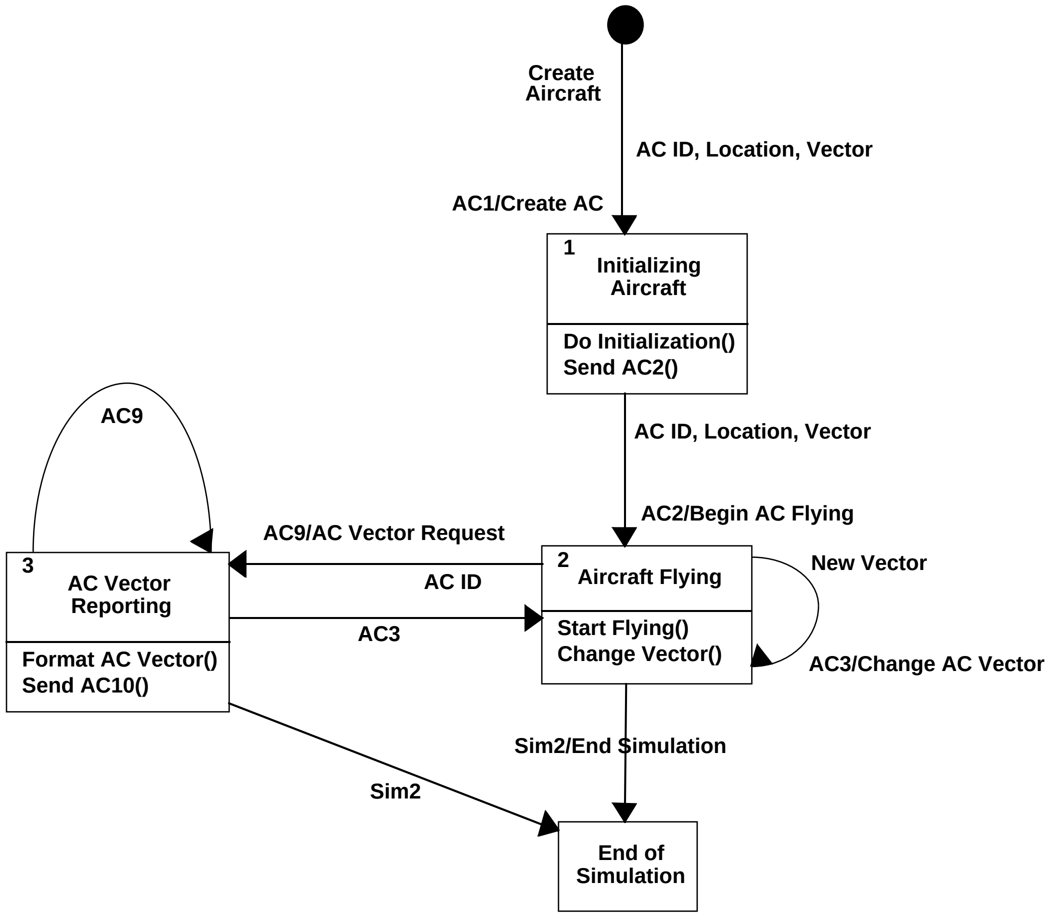


FIGURE 10. JMSS NP STD AIRCRAFT CLASS

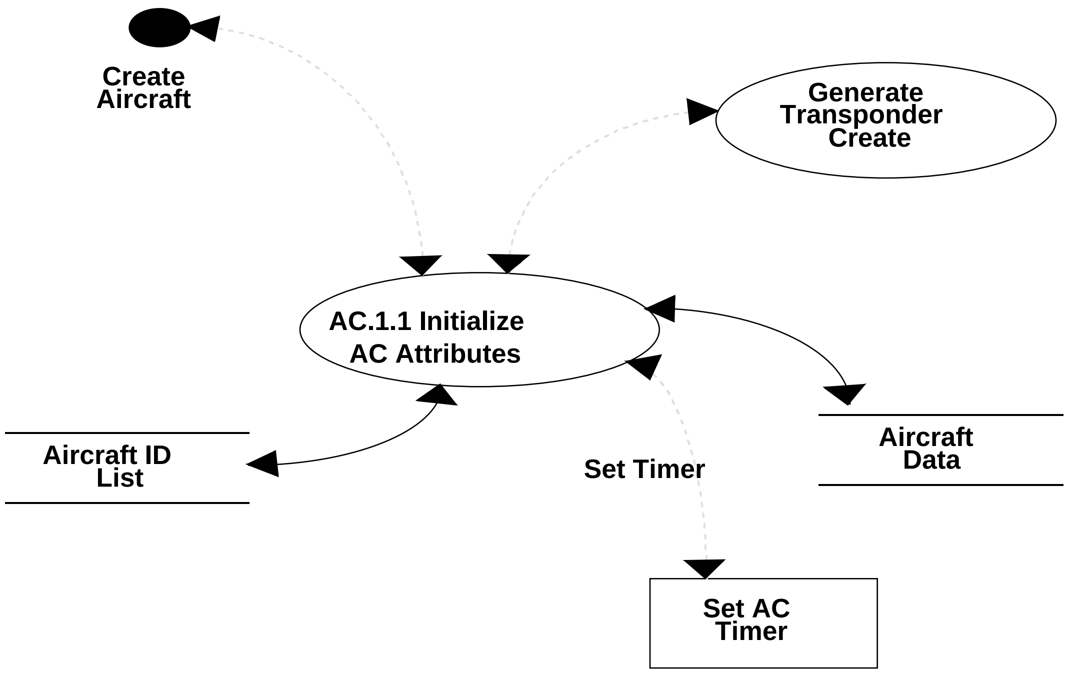


FIGURE 11. JMSS NP ADFD AIRCRAFT CLASS

<u>AC.1.1</u>
Create_Aircraft (Type, Position_Latitude, Position_Longitude)

Get Unique Acircraft_ID from Aircraft_ID_List

If Type = “Blue” then
    Maximum_Speed = 1032
    Minimum_Speed = 200
    Climb_Speed = 350
    Climb_Rate = 70
    Descent_Speed = 250
    Descent_Rate = -100
    Acel_a = 4.285*10-4
    Acel_b = -3.2*10-2
    Decel_a = -.175
    Decel_b = 6.273*10-3
    Current_Heading = 090
    Current_Speed = 450
    Current_Altitude = 25000
Else [Remark: Red Aircraft]
    Maximum_Speed = 1295
    Minimum_Speed = 200
    Climb_Speed = 300

```
        Climb_Rate = 65
        Descent_Speed = 275
        Descent_Rate = -150
        Acel_a = 3.26*10-4
        Acel_b = -2.8*10-3
        Decel_a = -.175
        Decel_b = 6.273*10-3
        Current_Heading = 270
        Current_Speed = 500
        Current_Altitude = 28000
Endif

Ordered_Heading = Current_heading
Ordered_Speed = Current_Speed
Ordered_Altitude = Current_Altitude
Create position update Trigger and set udate time to one minute
Time_of_Position = Current_Time
Write Data to Aircraft_Store
Generate AC2 (Creation and initialization complete => start flying)

End Create_Aircraft
```

<u>AC.1.2</u>
Generate the event to create a transponder

<u>AC.1.3</u>
Get current time; add one minute; create periodic timer.

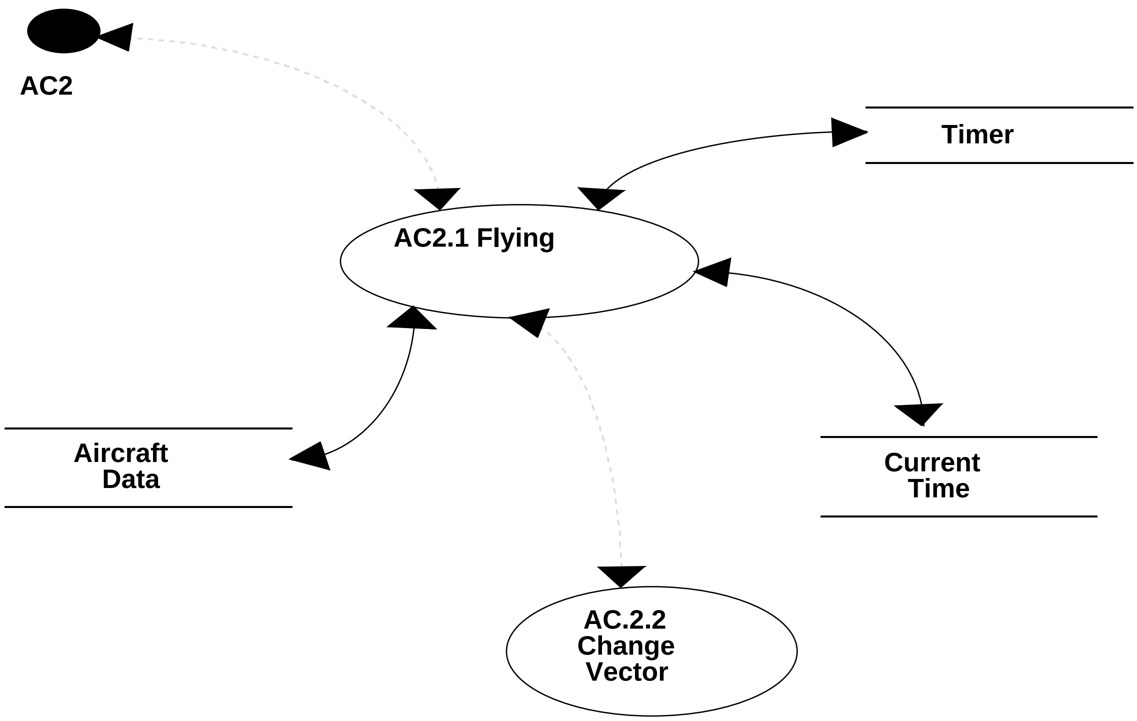


FIGURE 12. JMSS NP ADFD AIRCRAFT CLASS

AC.2.1

Timed Entry (Method Steps in Sequence):

A. Call Aircraft_Kinematics (Current_Heading, Ordered_Heading,Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude North_South_Distance, East_West_Distance)

B. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)

C. Time_of_Position = Current_Time.

D. Next_Update_Time = Current_Time plus one minute.

E. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude, Position_Latitude, Position_Longitude, Time_of_Position)

F. Reset timer

New Ordered_Heading Entry (Method Steps in Sequence):

Note: The procedure here is to compute current position based on the “old” ordered heading, speed, altitude and previous position time, and then “GET” from the operator’s command the “new” Ordered_Heading.

A. Call Aircraft_Kinematics (Current_Heading, Ordered_Heading,Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude North_South_Distance, East_West_Distance)

B1 Call Convert_Distances (North_South_Distance, East_West_Distance,

Position_Latitude, Position_Longitude)
B2. Get new Ordered_Heading
C. Call Direction_of_Turn (Ordered_Heading, Current_Heading, Turn_Direction, Degrees_of_Turn)
D. Degrees_of_Turn_Completed = Degrees_of_Turn[Remark: Aircraft turns]
E. Current_Direction_of_Turn = Turn_Direction[are "instantaneous."]
F. Time_of_Position = Current_Time.
G. Next_Update_Time = Current_Time plus one minute
H. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude, Position_Latitude, Position_Longitude, Time_of_Position)

New Ordered_Speed Entry (Method Steps in Sequence):
Note: The procedure here is to compute current position based on the "old" ordered heading, speed, altitude and previous position time, and then "GET" from the operator's command the "new" Ordered_Speed.

A. Call Aircraft_Kinematics (Current_Heading, Ordered_Heading,Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude North_South_Distance, East_West_Distance)
B. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)
C. GET new Ordered_Speed
D. If Ordered_Speed > Maximum_Speed then Ordered_Speed = Maximum_Speed
E. If Ordered_Speed < Minimum_Speed then Ordered_Speed = Minimum_Speed
F. Time_of_Position = Current_Time
G. Next_Update_Time = Current_Time plus one minute
H. Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude, Position_Latitude, Position_Longitude, Time_of_Position)

New Ordered_Altitude Entry (Method Steps in Sequence):
Note: The procedure here is to compute current position based on the "old" ordered heading, speed, altitude and previous position time, and then "GET" from the operator's command the "new" Ordered_Altitude.

A. Call Aircraft_Kinematics (Current_Heading, Ordered_Heading,Current_Speed, Ordered_Speed, Current_Altitude, Ordered_Altitude North_South_Distance, East_West_Distance)
B. Call Convert_Distances (North_South_Distance, East_West_Distance, Position_Latitude, Position_Longitude)
C. GET new Ordered_Altitude
D. If Ordered_Altitude > Maximum_Altitude then Ordered_Altitude = Maximum_Altitude
E. If Ordered_Altitude < Minimum_Altitude then Ordered_Altitude = Minimum_Altitude

```
F.      Time_of_Position = Current_Time
G.      Next_Update_Time = Current_Time plus one minute
H.      Export_for_Update (Current_Heading, Ordered_Heading, Current_Speed,
Ordered_Speed, Current_Altitude, Ordered_Altitude,
Position_Latitude, Position_Longitude, Time_of_Position)

Subroutine Aircraft_Kinematics (Current_Heading, Ordered_Heading,Current_Speed,
Ordered_Speed, Current_Altitude, Ordered_Altitude
                                        North_South_Distance, East_West_Distance)

Delta_Time_Min = Current_Time – Time_of_Position  [In MIN !!!]
Delta_Time_Sec = Current_Time – Time_of_Position  [In SEC !!!]

NS_Total = 0
EW_Total= 0
Delta_Time =  0166667 minutes [Remark: one second]
        If Ordered_Altitude > Current_Altitude then
        If Current_Speed = Climb_Speed then
                Call Constant_Heading_Distance (Current_Heading, Climb_Speed,
                        Delta_Time, North_South_Distance, East_West_Distance)
                NS_Total =  NS_Total + North_South_Distance
                EW_Total = EW_Total + East_West_Distance
                Call Aircraft_Heading_Change (Current_Heading, Ordered_Heading,
Climb_Speed)
                Current_Altitude = Current_Altitude + Climb_Rate
                If Current_Altitude >= Ordered_Altitude then
                        Current_Altitude = Ordered_Altitude
                Endif
        Else            [Remark: Aircraft must get to Climb_Speed before climbing]
                Call Aircraft_Speed_Change (Current_Speed, Climb_Speed,
Average_Speed)
                Call Aircraft_Heading_Change (Current_Heading, Ordered_Heading,
                                Average_Speed)
        Endif

Elseif Ordered_Altitude < Current_Altitude then
If Current_Speed = Descent_Speed then
Call Constant_Heading_Distance (Current_Heading, Descent_Speed,
Delta_Time, North_South_Distance, East_West_Distance)
NS_Total =  NS_Total + North_South_Distance
EW_Total = EW_Total + East_West_Distance
                Call Aircraft_Heading_Change (Current_Heading, Ordered_Heading,
Descent_Speed)
Current_Altitude = Current_Altitude + Descent_Rate
If Current_Altitude <= Ordered_Altitude then
        Current_Altitude = Ordered_Altitude
```

```
Endif

          Else                [Remark: Aircraft must get to Descent_Speed before Decending]
Call Aircraft_Speed_Change (Current_Speed, Descent_Speed,
Average_Speed)
                    Call Aircraft_Heading_Change (Current_Heading, Ordered_Heading,
                                                  Average_Speed)
          Endif

Elseif Current_Altitude = Ordered_Altitude[Remark: Aircraft regains Ordered_Speed]
          Call Constant_Heading_Distance (Current_Heading, Current_Speed,
                    Delta_Time, North_South_Distance, East_West_Distance)
          NS_Total =  NS_Total + North_South_Distance
          EW_Total = EW_Total + East_West_Distance
          Call Aircraft_Speed_Change (Current_Speed, Ordered_Speed,
Average_Speed)
          Call Aircraft_Heading_Change (Current_Heading, Ordered_Heading,
                                                  Average_Speed)
Endif

Delta_Time_Sec = Delta_Time_Sec – 1
Until Delta_Time_Sec <= 0

North_South_Distance = NS_Total
East_West_Distance = EW_Total

End Aircraft_Kinematics

Subroutine Aircraft_Heading_Change (Current_Heading, Ordered_Heading,
Average_Speed)
If Ordered_Heading <> Current_Heading then
Rate_of_Turn = 1091.5 / Average_Speed
          Current_Heading = Current_Direction_of_Turn * Rate_of_Turn +
                              Current_Heading
Degrees_of_Turn_Completed = Degrees_of_Turn_Completed –
Rate_of_Turn
If Degrees_of_Turn_Completed <= 0 then
          Current_Heading = Ordered_Heading
          Degrees_of_Turn_Completed = 0
Endif
Endif
End Aircraft_Heading_Change

Subroutine Aircraft_Speed_Change (Current_Speed, New_Speed, Average_Speed)

Old_Speed = Current_Speed
```

```
If New_Speed <> Current_Speed then
If New_Speed > Current_Speed then
Acceleration = 1 / (Acel_a* Current_Speed + Acel_b).
Current_Speed = Current_Speed + Acceleration
If Current_Speed > New_Speed then
        Current_Speed = New_Speed
Endif

        Elseif  New_Speed < Current_Speed then
                Deceleration = -.175 * EXP (.006273 * Current_Speed)
Current_Speed = Current_Speed + Deceleration
If Current_Speed < New_Speed then
        Current_Speed = New_Speed
Endif
Endif
Endif
Average_Speed = .5 (Old_Speed + Current_Speed)
End Aircraft_Speed_Change
```

Maximum_Speed (in knots to the nearest knot; e.g. 1030; HLA Real)
Minimum_Speed (in knots to the nearest knot; e.g. 200; HLA Real)
Climb_Speed (in knots to the nearest knot; e.g. 300; HLA Real)
Climb_Rate (in feet / second to the nearest foot; e.g. 70; HLA Real)
Decent_Speed (in knots to the nearest knot; e.g. 250; HLA Real)
Decent_Rate (in feet / second to the nearest foot; e.g. -75; HLA Real)
Current_Heading (in degrees to nearest degree; e.g. 243; HLA Real)
Ordered_Heading (in degrees to nearest degree; e.g. 283; HLA Real)
Current_Speed (in knots to the nearest knot; e.g. 15; HLA Real)
Ordered_Speed (in knots to the nearest knot; e.g. 25; HLA Real)
Current_Altitude (in feet to the nearest foot; e.g. 2400; HLA Real)
Ordered_Altitude(in feet to the nearest foot; e.g. 24000; HLA Real)
Position_Latitude (degrees to nearest second, North or South; e.g. 15-21-34N; HLA Real)
Position_Longitude (degrees to nearest second, East or West; e.g. 135-46-22W; HLA Real)
Time_of_Position (time to nearest minute; e.g. 1634; HLA Real)

3.2.3 RED AIRCRAFT OBJECT CLASS

There shall be only one Red Aircraft object. It shall also be an instance of the Aircraft Object Class. It shall "have an" IFF transponder capable of responding to IFF interrogations from ships, but only Modes 3/C (mode 3 code and altitude). Its behaviors shall be the same as the F-18C except with different values for its AC Object Class attributes.

3.2.4 IDENTIFICATION FRIEND OR FOE (IFF)

The IFF system shall consist of two objects: 1) the IFF interrogator/receiver, and 2) the IFF transponder. Only ship objects shall "have an" IFF interrogator/receiver. Ship and aircraft objects shall "have a" transponder. The STDs and ADFDs for both the Interrogator and the Transponder are described in this paragraph. In Build NP, to simplify processing, the transponder shall make the "within range" and horizon range calculations and reply, or not, depending on the outcome. In Build NP the maximum range for both the interrogator and transponder is 150 nm.

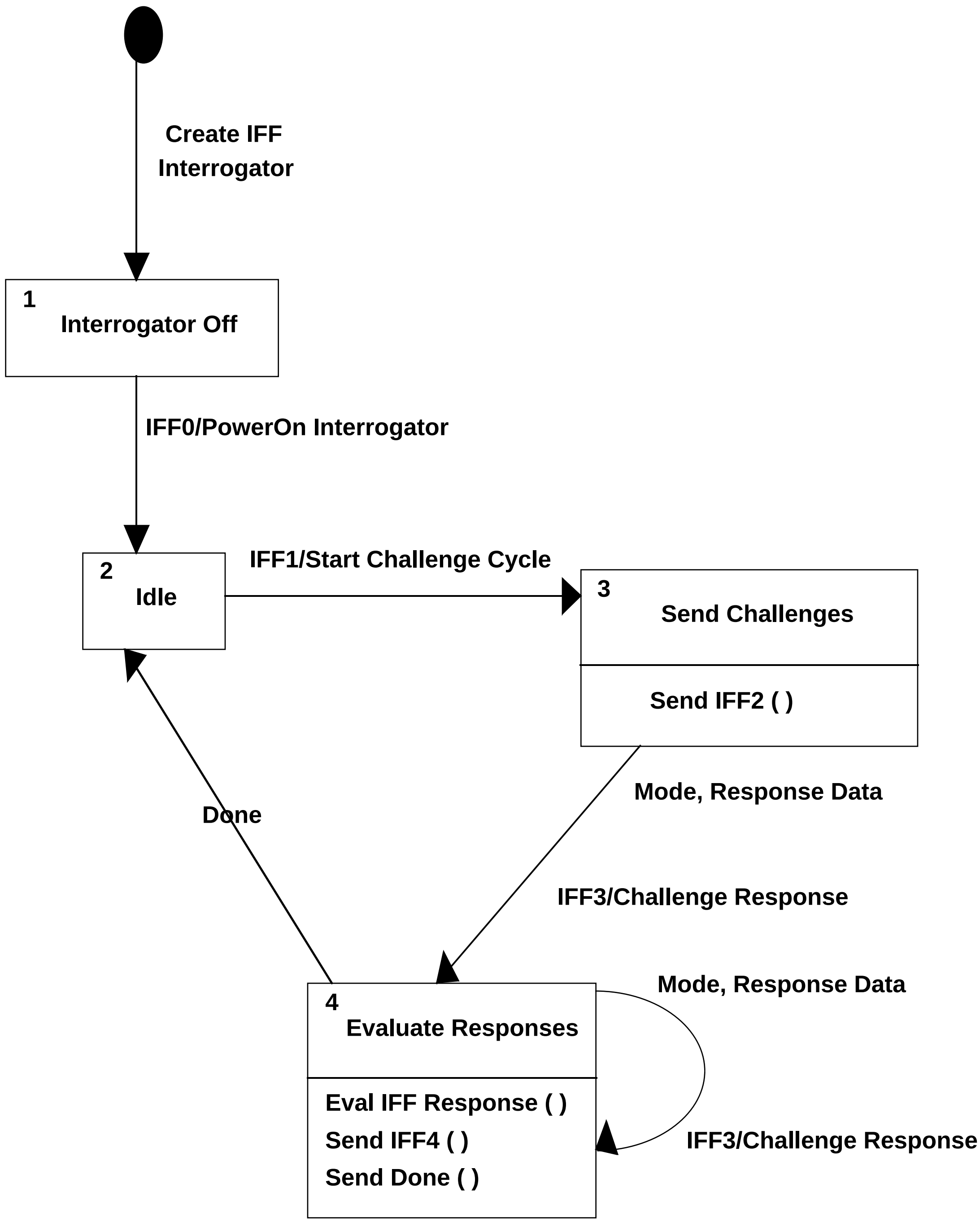


FIGURE 13. JMSS NP STD FOR THE IFF INTERROGATOR OBJECT CLASS

:

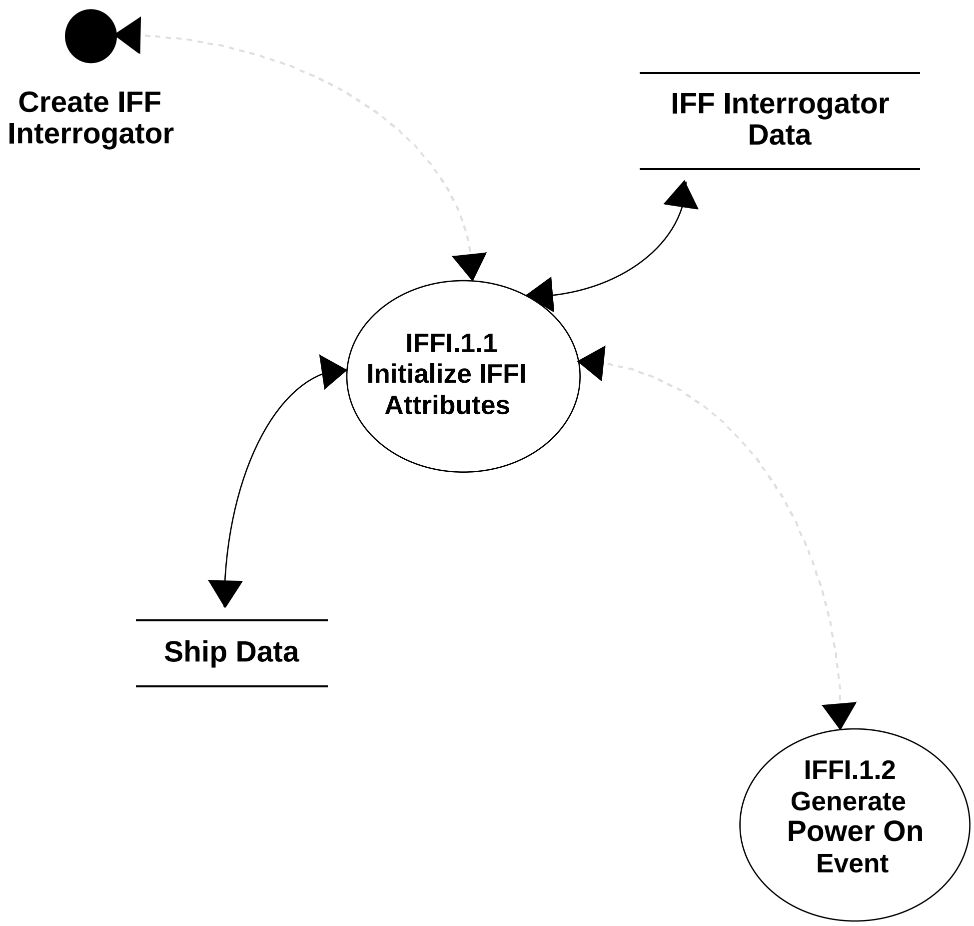


FIGURE 14. JMSS NP ADFD FOR THE IFF INTERROGATOR OBJECT CLASS

IFFI.1.1
A. Set I_State_Indicator to OFF
B. Next_Interrogation_Time = null
C. Export_for_Update (I_State_Indicator)
Attributes
I_State_Indicator (On /Off; e.g. On; Boolean)
    Reply_List HLA Complex Structure)
        Unique_Label (5 alphanumeric characters; e.g. 12Ab5; HLA Character)
        Bearing (in degrees to nearest degree; e.g. 243; HLA Real)
        Range (in nm to nearest nm; e.g. 121; HLA Real)

Item: Mode 1 code (two octal digits; e.g. 34; HLA Integer)
Item: Mode 2 code (four octal digits; e.g. 3427; HLA Integer)
Item: Mode 3/A code (four octal digits; e.g. 2356; HLA Integer)
Item: Mode 4 code ("YES" or null; e.g. YES; HLA String)
Item: Mode C aircraft altitude (in feet to nearest 100 feet; e.g 235; HLA Real)

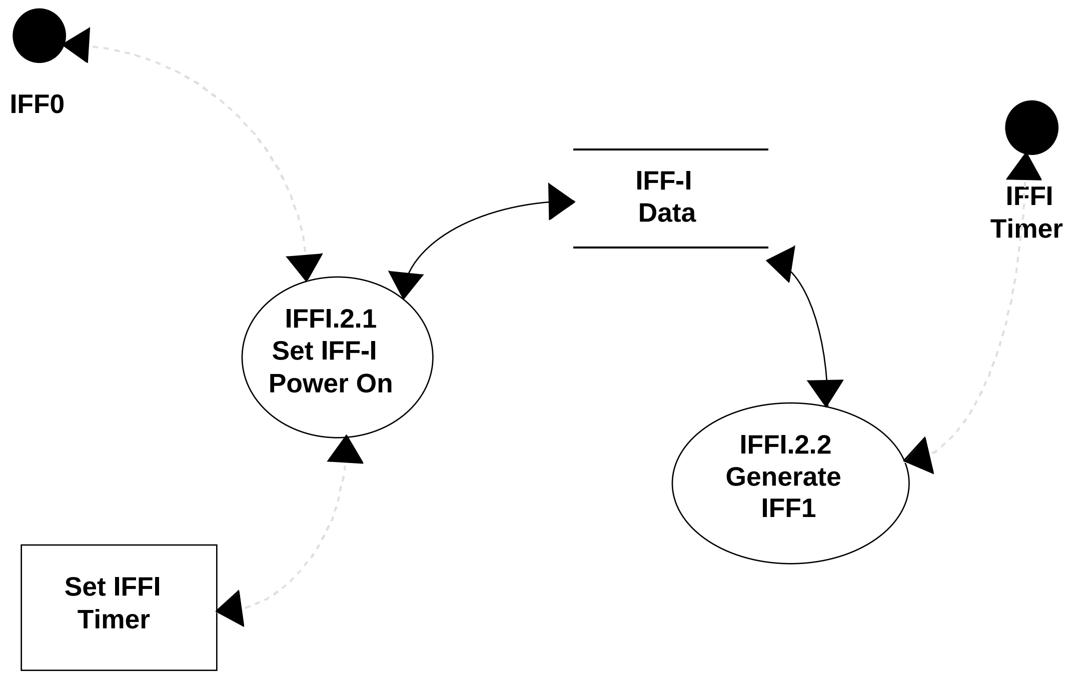


FIGURE 15. JMSS NP ADFD FOR THE IFF INTERROGATOR OBJECT CLASS

<u>IFFI.2.1</u>
Operator Turn_On Entry (Method Steps in Sequence)

A. Set I_State_Indicator to ON
B. Set I_Lat to host ship latitude (Position_Latitude)
C. Set I_Long to host ship longitude (Position_Longitude)
D. TRANSMIT (I_Lat, I_Long) interrogation to search volume.
E. Next_Interrogation_Time = Current_Time + one minute
F. Export_for_Update (I_State_Indicator, Reply_List)

<u>IFFI.2.2</u>
Timed_Entry (Method Steps in Sequence)

A. Set I-Lat to host ship latitude (Position_Latitude)
B. Set I_Long to host ship longitude (Position_Longitude)
C. TRANSMIT (I_Lat, I_Long) interrogation to search volume.

D.	Next_Interrogation_Time = Current_Time + one minute
E.	Export_for_Update (Reply_List)

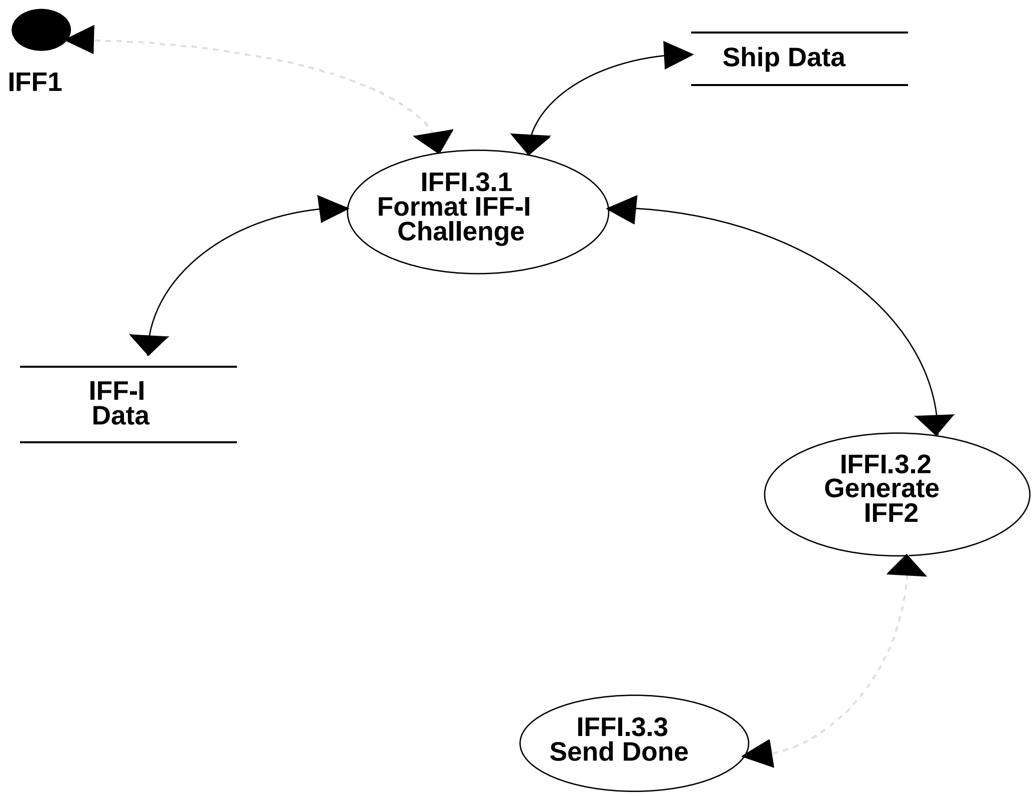


FIGURE 16. JMSS NP ADFD FOR THE IFF INTERROGATOR OBJECT CLASS

<u>IFFI.3.1</u>
A.	Set I-Lat to host ship latitude (Position_Latitude)
B.	Set I_Long to host ship longitude (Position_Longitude)
C.	TRANSMIT (I_Lat, I_Long) interrogation to search volume.
D.	Call Receive_Replies
E.	Next_Interrogation_Time = Current_Time + one minute

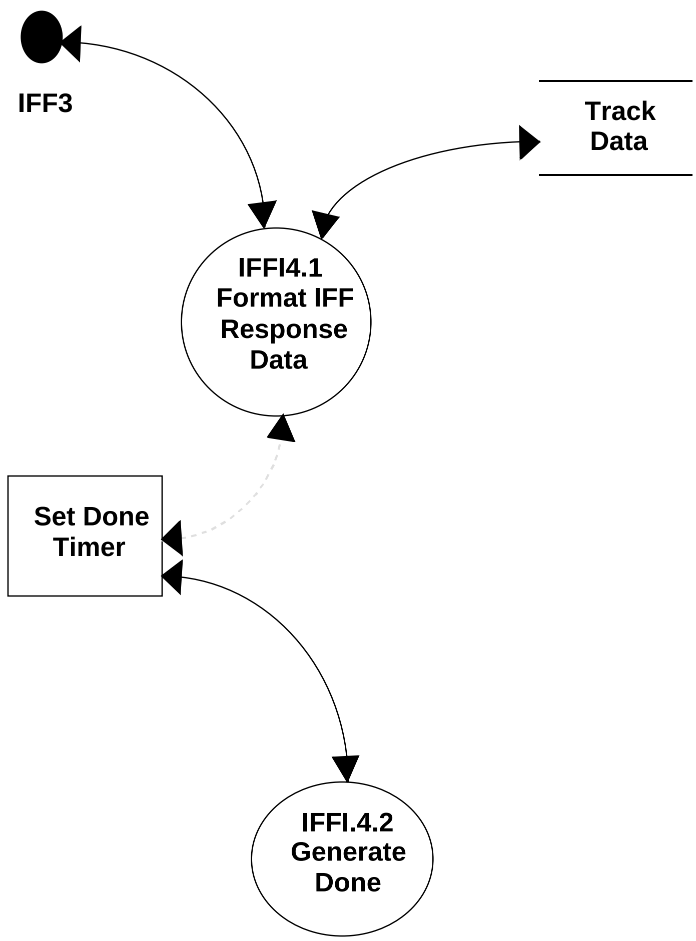


FIGURE 17. JMSS NP ADFD FOR THE IFF INTERROGATOR OBJECT CLASS

IFFI.4.1
Receive_Replies

"Build" Reply_List (one entry for each replying track)
Item: Unique Label
Item: Bearing from Interrogator to replying track
Item: Range from interrogator to replying track.
Item: Mode 1 code
Item: Mode 2 code
Item: Mode 3/A code
Item: Mode 4 code

Item: Mode C aircraft altitude
End Receive_Replies

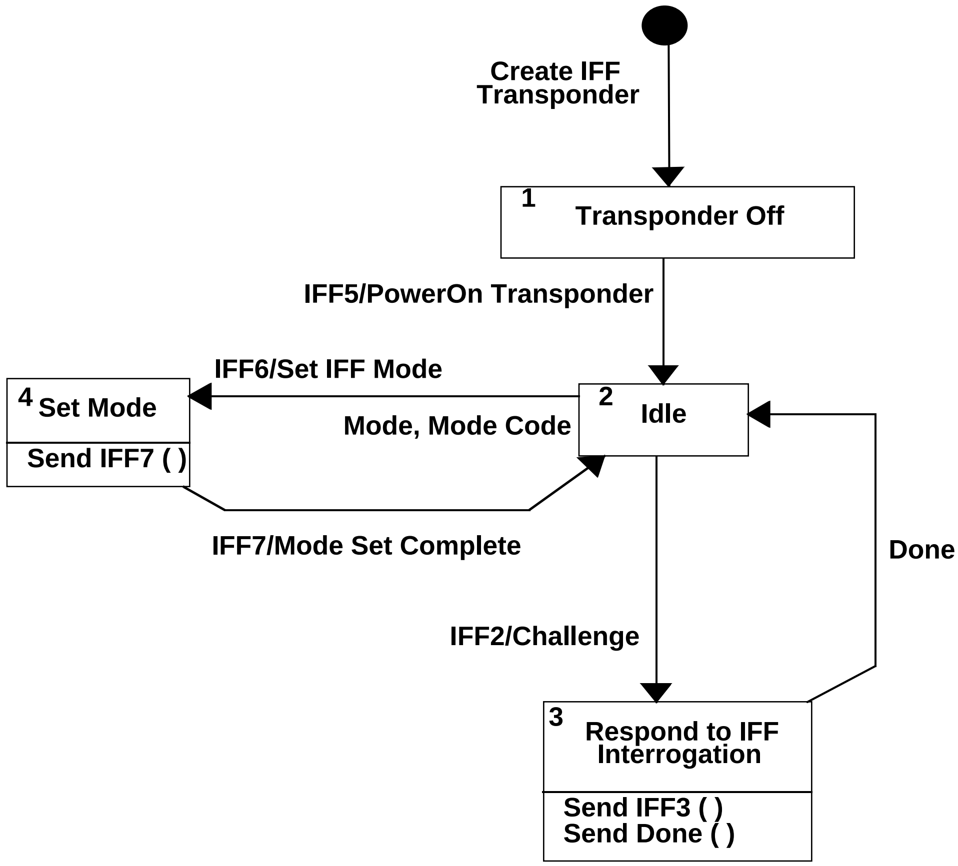


FIGURE 18. JMSS NP STD IFF TRANSPONDER OBJECT CLASS

The coded pulse information provided by a transponder are associated with five modes and their associated codes:

Mode 1	Two digit octal code
Mode 2	Four digit octal code
Mode 3/A	Four digit octal code
Mode C	Aircraft altituded (-1,000 feet to + 126,000 feet)
Mode 4	Military encrypted identification

IFF is principly used to identify tactical aircraft. Pilots provide information by manipulating the modes and codes of their aircraft transponders. Modes 1 and 3/A codes may be modified in flight by the pilot. Modes 2, 4 and C may not be modified by the pilot. However, pilots can turn any individual mode on and off, as well as turning the entire transponder off.

Mode 4 is the only mode which provides positive identification since both military (sans mode 4) and commercial transponders have been sold throughout the world to many different purchasers.

The content of the coded replies enables the operators to assess the identity of the responding aircraft. If military, many specific codes have meanings associated with various units, activities, missions. The mode 2 code, for example, is the address used by an aircraft carrier's Automatic Carrier Landing System (ACLS) to ensure signals are received by the correct aircraft. The maximum range for both the transponder and interrogator is 150 nm. This is the number to be used in the within range test.

:

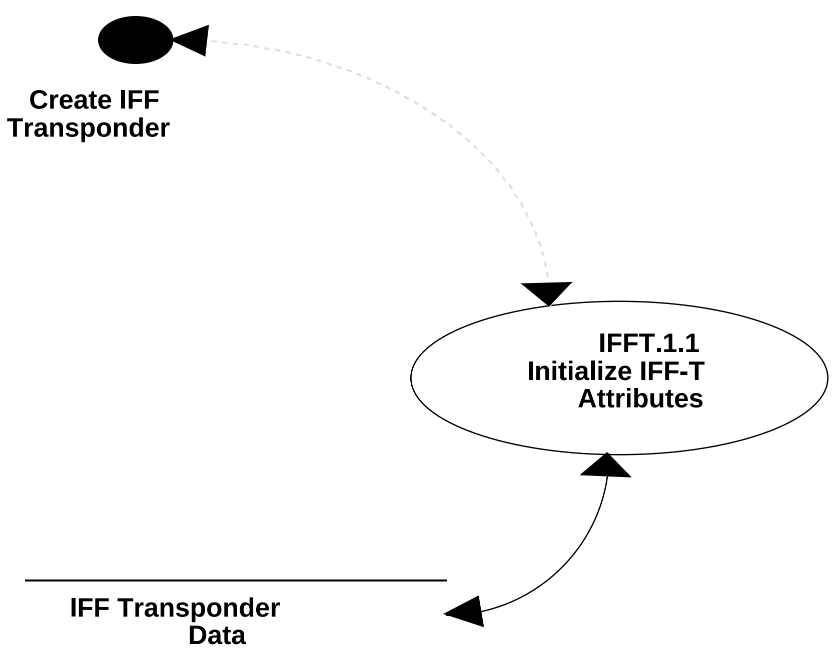


FIGURE 19. JMSS NP ADFD IFF TRANSPONDER OBJECT CLASS

<u>IFFT.1.1</u>
Create IFF Transponder Process

Create_IFF_Transponder (Type)

GET Transponder_ID from Transponder_ID_List

```
        T_State_Indicator = “Off”
        Mode_1_OnOff_Indicator = “On”
        Mode_2_OnOff_Indicator = “On”
        Mode_3/A_OnOff_Indicator = “On”
        Mode_4_OnOff_Indicator = “On”
        Mode_C_OnOff_Indicator = “On”
        Mode_1_Code= 01
        Mode_2_Code= 2345
        Mode_3/A_Code= 6701
        Mode_4_Code= “Yes”
        Mode_C_Code= 0123
        Reply_List = “null”

        Case Type Is
                Type = “Red”
                        Mode_1_OnOff_Indicator = “Off”
                        Mode_2_OnOff_Indicator = “Off”
                        Mode_4_OnOff_Indicator = “Off”
                        Mode_4_Code= “No”
                Type = “Ship”
                        Mode_C_OnOff_Indicator = “Off”
                Type = “All Others”
                        null;
        End Case;

        Write Attributes to Transponder_Store
                *Transponder_ID
                Type
                T_State_Indicator
                Mode_1_OnOff_Indicator
                Mode_2_OnOff_Indicator
                Mode_3/A_OnOff_Indicator
                Mode_4_OnOff_Indicator
                Mode_C_OnOff_Indicator
                Mode_1_Code
                Mode_2_Code
                Mode_3/A_Code
                Mode_4_Code
                Mode_C_Code
                Reply_List
        End Write

End Create_IFF_Transponder
```

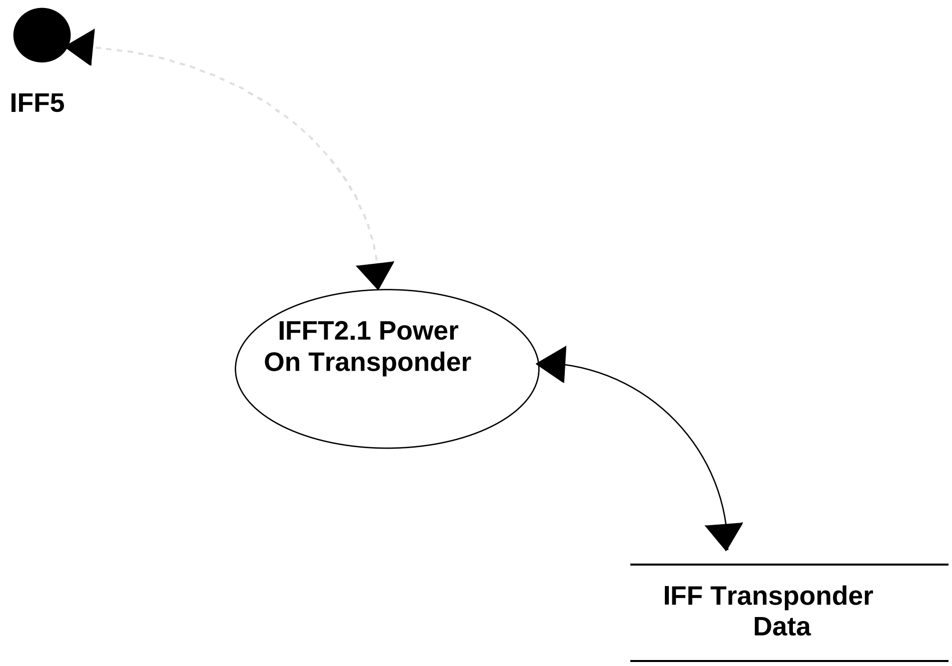


FIGURE 20. JMSS NP ADFD IFF TRANSPONDER OBJECT CLASS

<u>IFFT.2.1</u>
When on, any individual mode the transponder possess shall be selectable to on or off. A mode which is on shall have its code included in the transpoder's reply; one which is off shall not. If the transponder is on but all modes selected off then the transponder shall not reply to the interrogation in spite of the fact that it is "on".

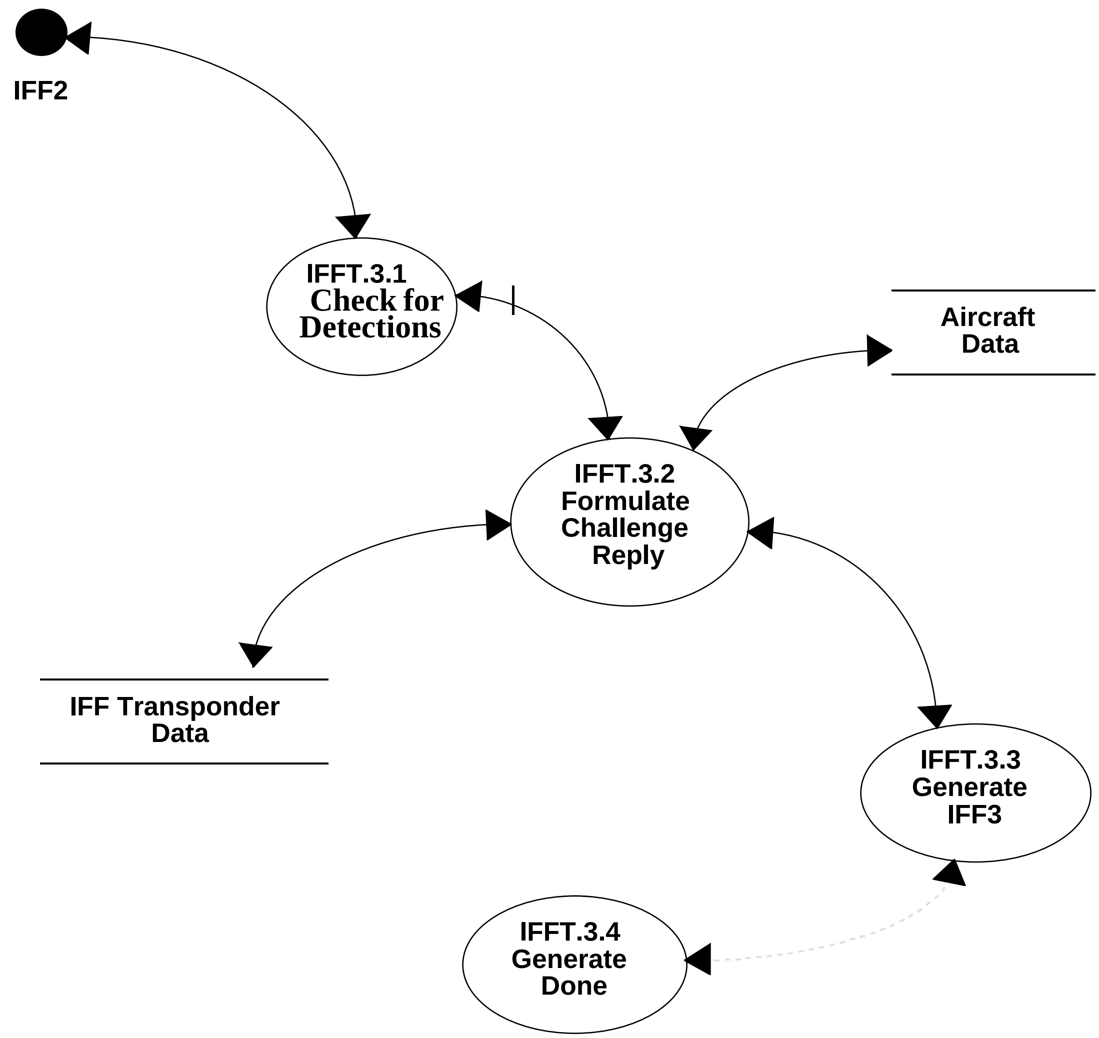


FIGURE 21. JMSS NP ADFD IFF TRANSPONDER OBJECT CLASS

<u>IFFT.3.1</u>
In Build NP the maximum range for both the transponder and interrogator is 150 nm. This is the number to be used in the with-in range test.

In Build NP the radar horizon range equation shall be in nautical miles and equal to
1.23*[Square_Root (IFF_Antenna_Height + Square_Root (Track_Altitude)]

In Build NP the IFF_Antenna_Height shall be the Generic Ship Hull height above the water line , 136 feet. Square_Root (136) = 11.66. Thus

Horizon_Range = 1.23 * Square_Root (Track_Altitude) + 14.34.

Interrogation Entry (Method Steps in Sequence)

```
A.      M_State_Indicator = Off
        For each Mode_(i) = On[Remark: At least one mode must]
                M_State_Indicator = On[be on for a reply to be sent. Note:]
        Endfor                                  [If present, Mode 4 is always on.]
B.      If T_State_Indicator = Off OR M_State_Indicator = Off then
                Exit Method                 [Remark: No reply if transponder or]
        Endif                                   [all modes are off.]
C.      Get I_Lat and I_Long from interrogation message
D.      Set T_Lat = Transponder's host Position_Latitude
E.      Set T_Long = Trandponders's host Position_Longitude
F.      Call Bearing_Range (I_Lat, I_Long, T_Lat, T_Long, Bearing, Range)
G.      If Transonder's host = Ship then
                Set Track_Altitude = 136[Remark: Ship's height above water line]
        Else
                Set Track_Altitude = Current_Altitude
        Endif
F.      Horizon_Range = 1.23 * Square_Root (Track_Altitude) + 14.34.
G.      If Range <= 150 AND Range <= Horizon_Range then[Remark: With-in]
                Call Reply_to_Interrogation   [range and horizon]
        Endif                                           [range checks.]
```

IFFT.3.2

Subroutine Reply_to_Interrogation

```
Construct Reply_Message
        Bearing
        Ranget
        For each Mode_(i) = ON
                Set Item(i) = Mode_(i)_Code
        Endfor
End Construct Reply_Message
Send Reply_Message to Interrogator
End Reply_to_Interrogation
```

Subroutine Bearing_Range (I_Lat, I_Long, T_Lat, T_Long, Bearing, Range)

Note: Bearing is based on true north; i.e. no variation. Navigation is by Rhumb line. Bearing and range calculations shall be based on the map display functionality chosen by the SWEC IPT to support Build NP and current positions.

```
Bearing = Nav_Function (I_Lat, I_Long, T_Lat, T_Long)
Range = Nav_Function (I_Lat, I_Long, T_Lat, T_Long)

End Bearing_Range
```

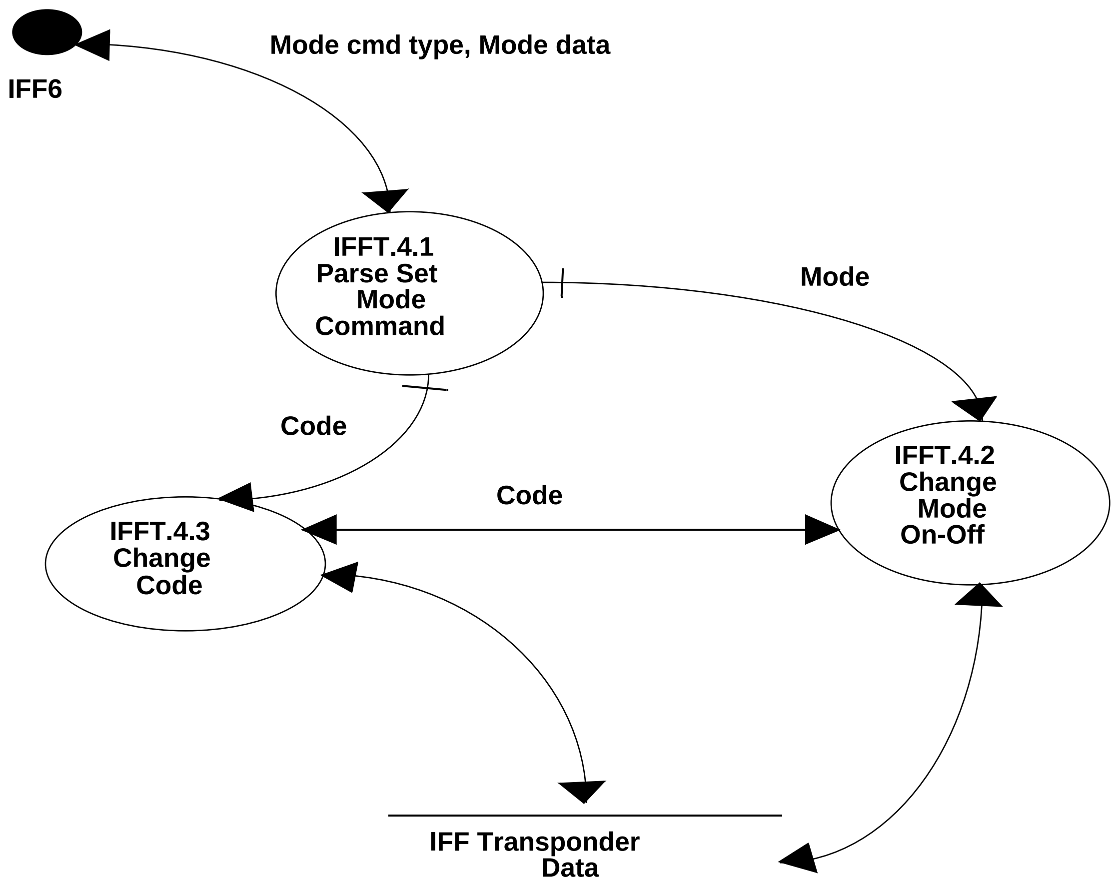


FIGURE 22. JMSS NP ADFD IFF TRANSPONDER OBJECT CLASS

<u>IFFT.4.2</u>
Mode_OnOff_Code

GET Type from Transponder_Store

GET Mode_ID, OnOff_Indicator and Code from IFF6 Set IFF Mode message

```
Case Mode_ID Is
          Case Mode_ID = "1"
If Type = "Blue" OR Type = "Ship" then
                              If OnOff_Indicator <> null then
                                        Mode_1_Indicator = OnOff_Indicator
                    Write Mode_1_Indicator to Transponder_Store
                              Endif
```

```
                        If Code <> null then
                                Mode_1_Code = Code
                        Write Mode_1_Code to Transponder_Store
                        Endif
                Endif
        Case Mode_ID = “2”
If Type = “Blue” OR Type = “Ship” then
                        If OnOff_Indicator <> null then
                                Mode_2_Indicator = OnOff_Indicator
                        Write Mode_2_Indicator to Transponder_Store
                        Endif
                        If Code <> null then
                                Mode_2_Code = Code
                        Write Mode_2_Code to Transponder_Store
        Endif
Endif
        Case Mode_ID = “3/A”[Remark: Everybody has mode 3/A.]
                If OnOff_Indicator <> null then
                        Mode_3/A_Indicator = OnOff_Indicator
                Write Mode_3/A_Indicator to Transponder_Store
                Endif
                If Code <> null then
                        Mode_3/A_Code = Code
                Write Mode_3/A_Code to Transponder_Store
                Endif
        Case Mode_ID = “C”
                If Type = “Blue” OR Type =“Red”
If Mode_ID = “C” AND OnOff_Indicator <> null then
                                Mode_C_Indicator = OnOff_Indicator
                        Write Mode_C_Indicator to Transponder_Store
                        Endif
                Endif
        Case Mode_ID = “All_Others”[Remark: Can’t change Mode 4 in any way in Build NP.]
                null;
End Case

Generate IIF7 (Mode Set Complete)

End Mode_OnOff_Code
```

3.2.5 TRACK OBJECT

The track object class is used to perfect the association between the ship and the Aircraft

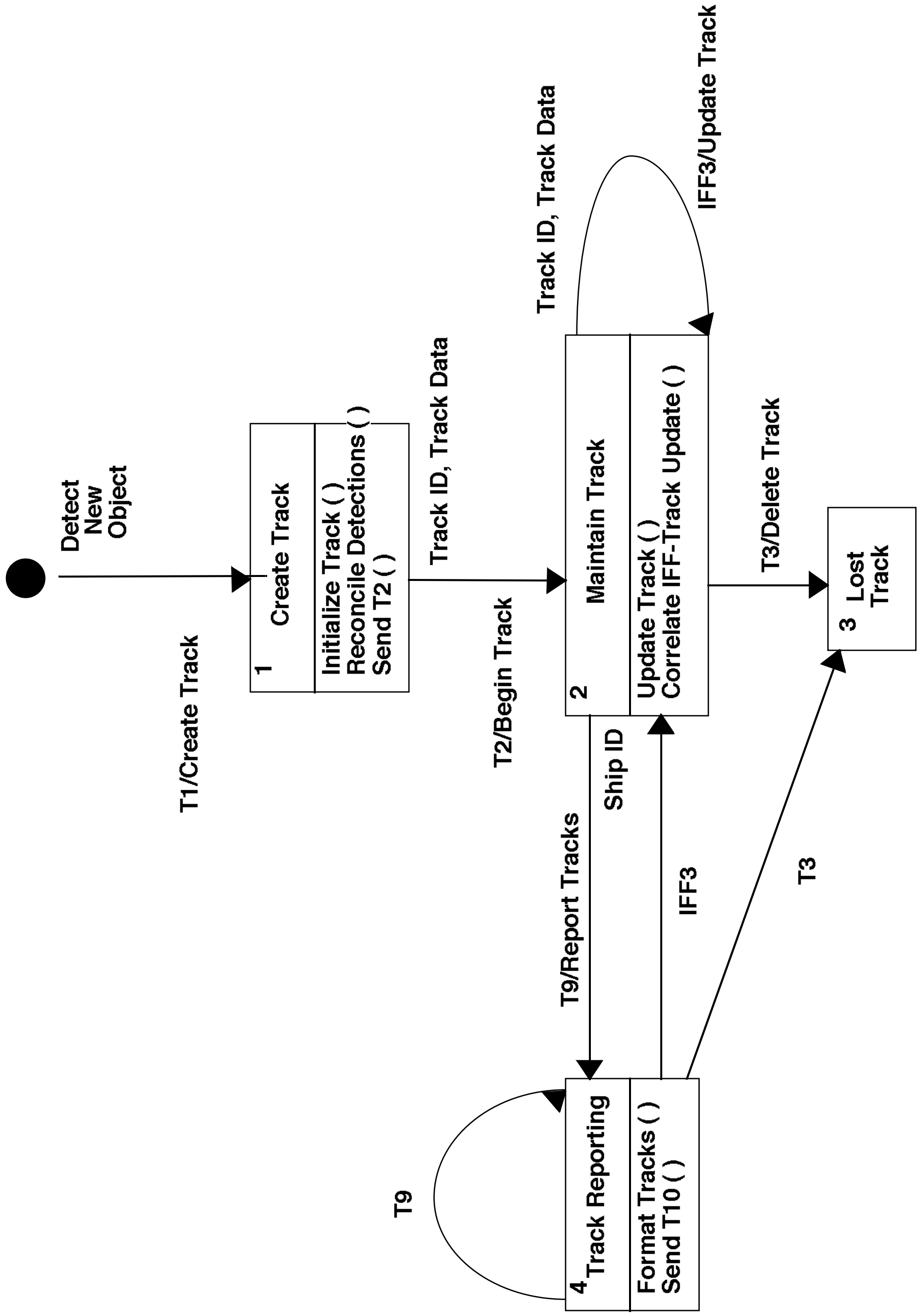


FIGURE 23. JMSS NP STD TRACK OBJECT CLASS

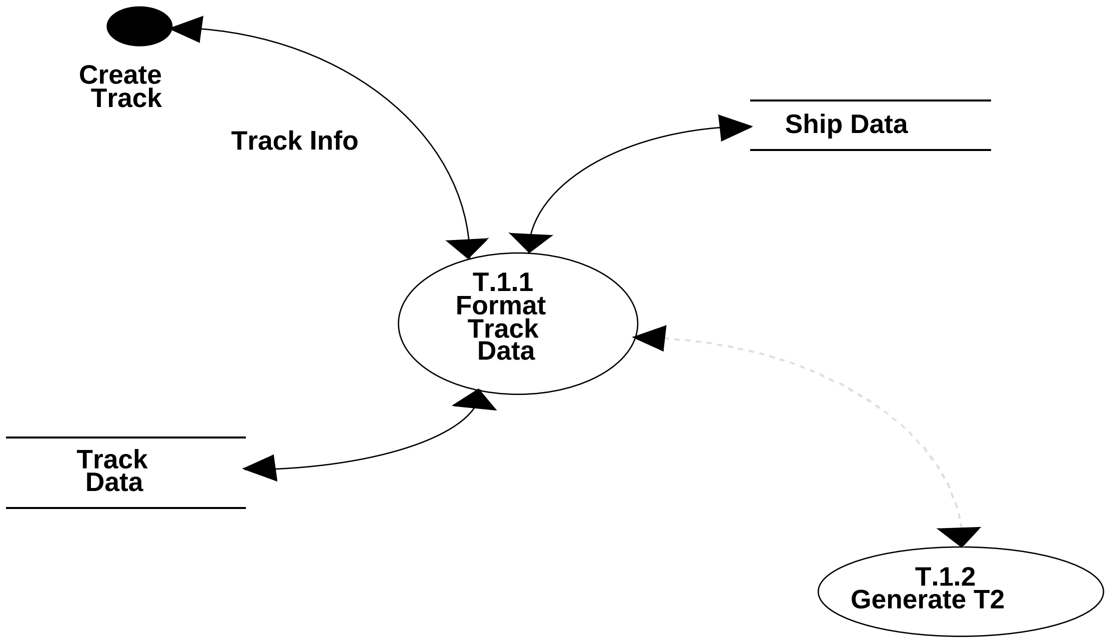


FIGURE 24. JMSS NP ADFD TRACK OBJECT CLASS

<u>T.1.1</u>

Create List; data stored

Unique_Label Transponder id (5 alphanumeric characters; e.g. 12Ab5; HLA Character)
Bearing (in degrees to nearest degree; e.g. 243; HLA Real)
Range (in nm to nearest nm; e.g. 121; HLA Real)
Item: Mode 1 code (two octal digits; e.g. 34; HLA Integer)
Item: Mode 2 code (four octal digits; e.g. 3427; HLA Integer)
Item: Mode 3/A code (four octal digits; e.g. 2356; HLA Integer)
Item: Mode 4 code ("YES" or null; e.g. YES; HLA String)
Item: Mode C aircraft altitude (in feet to nearest 100 feet; e.g 235; HLA Real)
Target id (5 alphanumeric characters; e.g. 12Ab5; HLA Character)
Update time (minutes; HLA Character)

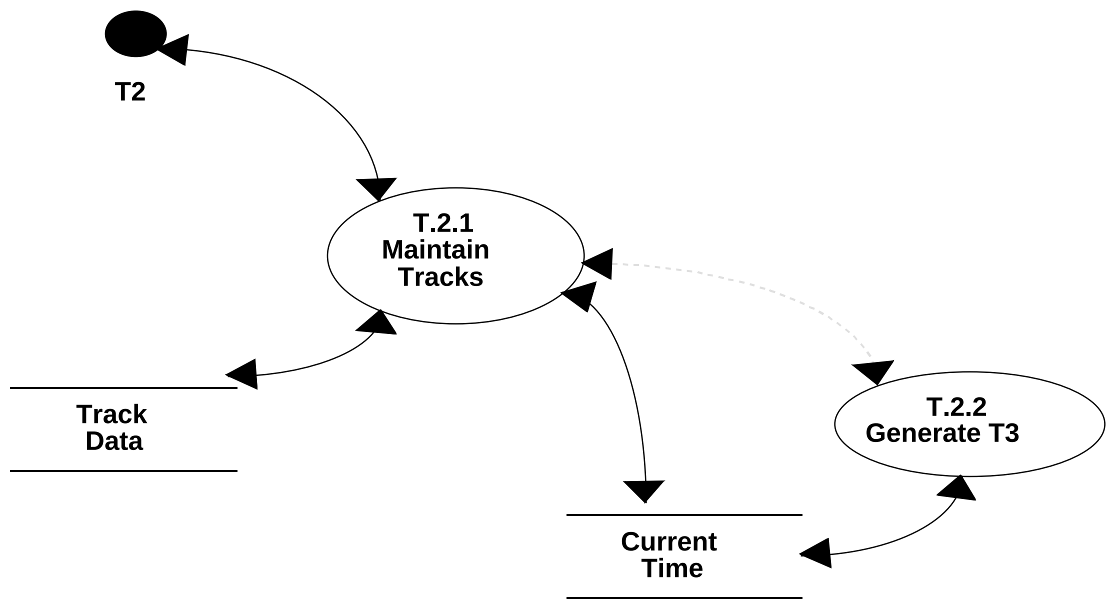


FIGURE 25. JMSS NP ADFD TRACK OBJECT CLASS

T.2.1
A. Receive event to update track with track data.
B. Get corresponding track data from Track Data store.
C. Get Current Time.
D. Update track data and write back to data store.
E. Call "T.2.2".

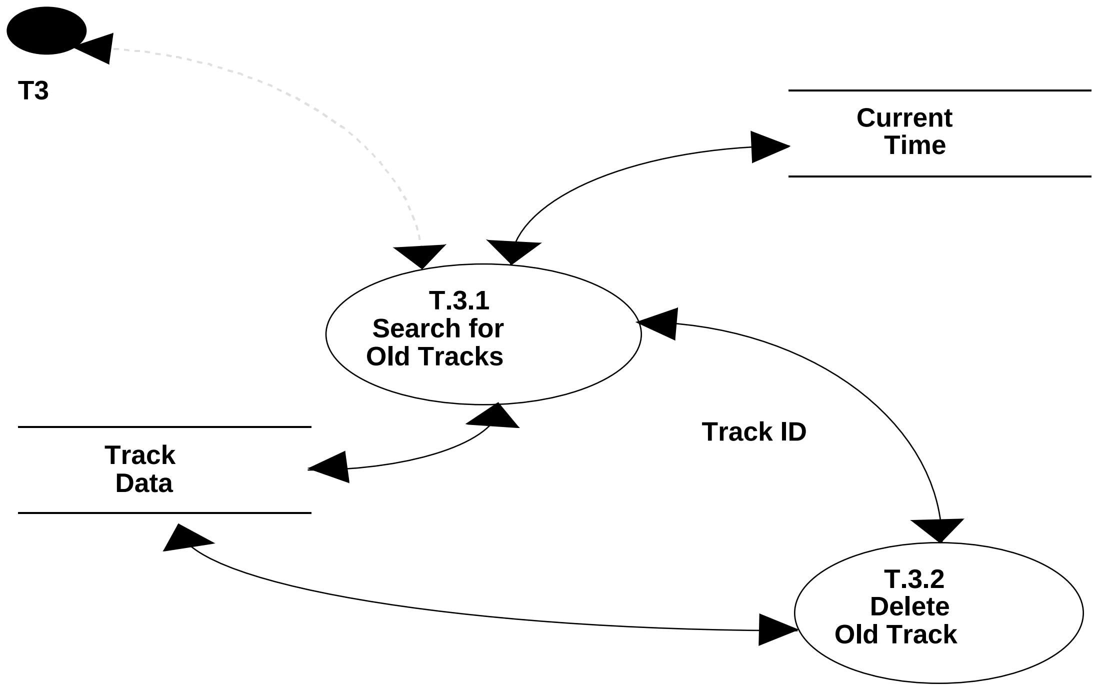


FIGURE 26. JMSS NP ADFD TRACK OBJECT CLASS

<u>T.3.1</u>

```
Search for_old_tracks
Check_time = current_time - 5 minutes
For each track in Track_list
        If update_time < check_time then
                delete_track
        endif
end for
```

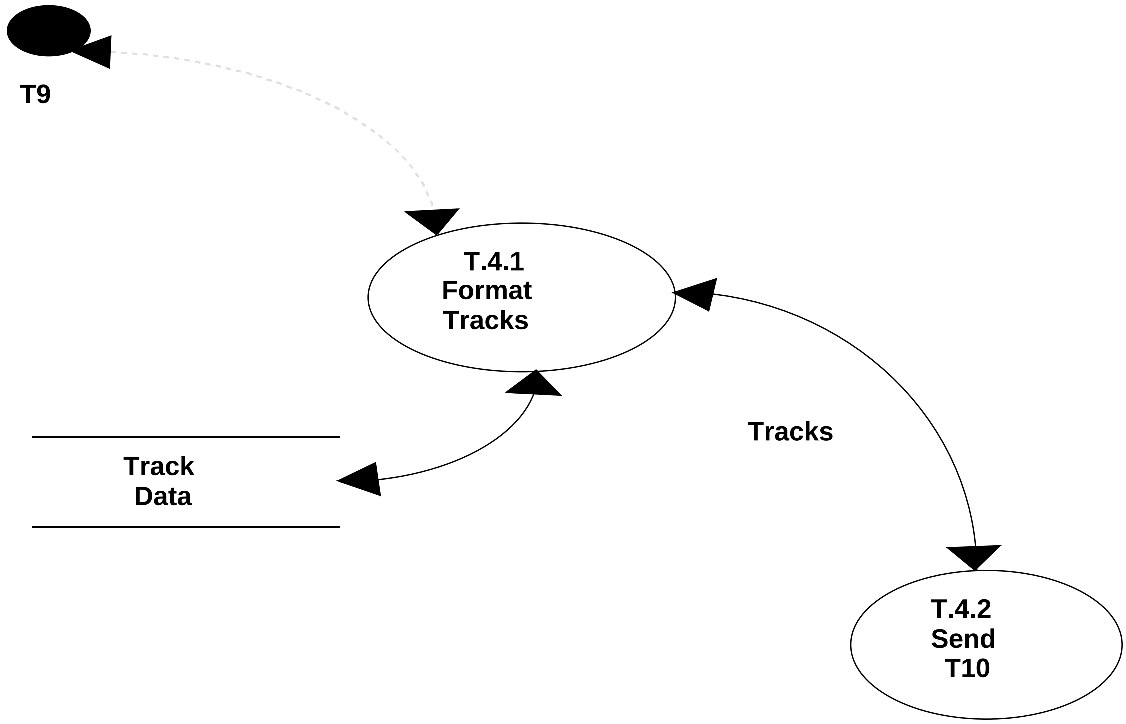


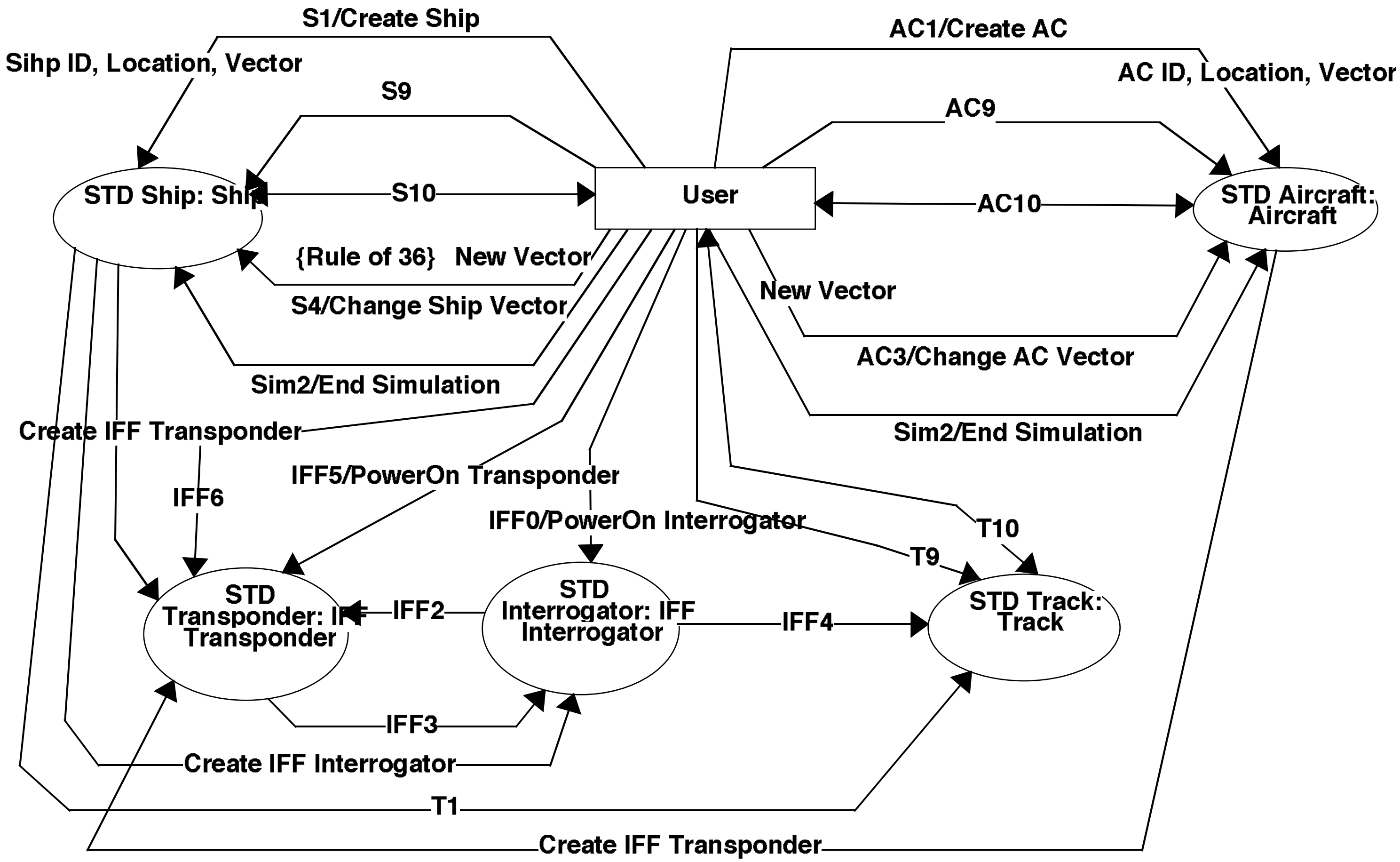
FIGURE 27. JMSS NP ADFD TRACK OBJECT CLASS

<u>T.4.1</u>
A. Receive event to report tracks.
B. Get track data from Track Data store.
C. Format track data.
D. Send data to "T.4.2 Send T10".

3.3 JMSS NP EXTERNAL/INTERNAL INTERFACE REQUIREMENTS.

The Human Computer Interface (HCI) shall be implemented as a single work station in the fashion of a Surrogate C4I System. Beyond the HCI display there are no other external interface requirements or specifications. The user operates through the HCI. In this case the users are the testors.

NP OCM



S1/Create Ship
Sihp ID, Location, Vector
S9
STD Ship: Ship
S10
User
{Rule of 36} New Vector
S4/Change Ship Vector
Sim2/End Simulation
Create IFF Transponder
IFF6
IFF5/PowerOn Transponder
IFF0/PowerOn Interrogator
AC1/Create AC
AC ID, Location, Vector
AC9
AC10
STD Aircraft: Aircraft
New Vector
AC3/Change AC Vector
Sim2/End Simulation
T10
T9
STD Transponder: IFF Transponder
IFF2
STD Interrogator: IFF Interrogator
IFF4
STD Track: Track
IFF3
Create IFF Interrogator
T1
Create IFF Transponder

FIGURE 28. JMSS NP OBJECT COMMUNICATIONS MODEL (OCM)

The above figure identifies the external interfaces in the system. These interfaces shall be specified at the Application Layer of the ISO Model for Open Systems Architecture. The Session, Transport, Network and Physical layers are implemented in the operating system software and the hardware devices. These interface definitions are advisory only. The HCI is used to both unit test and to integration test. A Software Design Description (SDD) Appendix to specify the HCI to be used and built for testing and demonstration purposes. The operational HCI will be supplied by JSIMS Joint Project Office (JPO).

The IDs of the interfaces are:

- SM-HCI Simulation Model (SM) of the JMSS NP to Human Computer Interface (HCI) for the JMSS NP
- HCI-U Human Computer Interface (HCI) of the JMSS NP to the User (U)

3.3.1 SM-HCI INTERFACE

3.3.1.1 DATA TO THE HCI FROM THE SM

1 .Position Data

This data shall contain the identification, position, course, speed and altitude of each object having data sent to the HCI. This shall be data designated by the HCI to be of interest. It shall be the responsibility of the HCI to assign the proper NTDS icon to the track and to display the position report in the proper place on the screen. This display shall be against a geographical display that shall be created using the Map Draw Module of Caricature.(TM) With the passage of time, the HCI shall compute new positions and display same. The frequency shall be contained in the design for the HCI. This data transfer shall occur when an event happens affecting the object. This data shall be supplied only for those items requested by the HCI to be of interest.

1 .Textual Data

Textual data shall be status information upon the request of the HCI. This shall be tabular information about every object in the track database. It shall contain the same position information as represented above.

3.3.1.2 DATA TO THE SM FROM THE HCI

1 .Position Data

None

1 .Textual Data

This data shall contain the entities to be instantiated at start-up and their initial points; a list.

3.3.2 HCI-U INTERFACE

3.3.2.1 DATA TO THE U FROM THE HCI

1 .Position Data

This class of data shall be displayed using the Map Draw Module in Geographical form. NTDS icons shall be used for presentation. Each report shall include ID, position, course and speed. This data, other than iconic presentation, shall be displayed upon requested. The tracks shall be those previously requested by the user. Position data shall be produced upon the occurrence of an event in the SM. These reports shall be unsolicited.

1 .Textual Data

Status reports shall be presented for each track. These data shall include the position of each track in the SM. These track reports shall be updated upon the occurrence of an event. The reports shall be displayed only upon request. They must be solicited.

3.3.2.2 DATA TO THE HCI FROM THE U

1 .Position Data

None

1 .Textual Data

The initial list of platforms and their position, course and speed. This shall be in list form. A second list shall be that of tracks of interest. This shall be the tracks to be displayed on the Geo Tactical Display. Also, requests to change various attributes.

3.4 JMSS NP INTERNAL INTERFACE REQUIREMENTS.

All additional internal interface requirements will be the subject of the Software Design Description (SDD).

3.5 JMSS NP INTERNAL DATA REQUIREMENTS.

All internal data shall be the subject of the Software Design Description (SDD).

3.6 ADAPTATION REQUIREMENTS.

This paragraph is tailored out.

3.7 SAFETY REQUIREMENTS.

No special requirements

3.8 SECURITY AND PRIVACY REQUIREMENTS.

No special requirements

3.9 JMSS NP ENVIRONMENT REQUIREMENTS.

No special requirements

3.10 COMPUTER RESOURCE REQUIREMENTS.

3.10.1 COMPUTER HARDWARE REQUIREMENTS.

The hardware to be used for the development and testing of the JMSS NP are contained in the Laboratory in Bldg. 606 Room 241A. The machines are designated "Pepper" and "Coke".

3.10.2 COMPUTER HARDWARE RESOURCE UTILIZATION REQUIREMENTS.

There are no special requirements.

3.10.3 COMPUTER SOFTWARE REQUIREMENTS.

The existing Map Draw Module of the Caricature software system shall be used for the Geo-Tactical display. IMPORT shall be used as the design translation software to generate the C++ code for the JMSS NP .

3.10.4 COMPUTER COMMUNICATIONS REQUIREMENTS.

The Local Area Network is in place to support the development. No other special requirements exist.

3.11 SOFTWARE QUALITY FACTORS.

Software quality assurance shall be limited to inspection of products by the SWEC staff. Appropriate requirements and design reviews shall be held by the Build Manager as called for in the POA&M>

3.12 DESIGN AND IMPLEMENTATION CONSTRAINTS.

None

3.13 PERSONNEL-RELATED REQUIREMENTS.

None

3.14 TRAINING-RELATED REQUIREMENTS.

None

3.15 LOGISTICS-RELATED REQUIREMENTS.

None

3.16 OTHER REQUIREMENTS.

None

3.17 PACKAGING REQUIREMENTS.

None

3.18 PRECEDENCE AND CRITICALITY OF REQUIREMENTS.

None

4.0 QUALIFICATION PROVISIONS.

This section defines a set of qualification methods and specifies, for each software requirement (State) in Section 3, the method(s) to be used to ensure that the requirement has been met. A table is used to present this information. Qualification methods include:

- a. Demonstration: The operation of the JMSS NP , or a part of the JMSS NP , that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
- b. Test: The operation of the JMSS NP , or a part of the JMSS NP , using instrumentation or other special test equipment to collect data for later analysis.
- c. Analysis: The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpretation, or extrapolation of test results.
- d. Inspection: The visual examination of JMSS NP code, documentation, etc.
- e. Special qualification methods: Any special qualification methods for the JMSS NP , such as special tools, techniques, procedures, facilities, and acceptance limits.

Table 1: Qualification Requirements

States	Demonstration	Test	Analysis	Inspection	Special Qualification
Figure 5 State 1	X	X			
State 2		X	X		
State 3		X	X		
State 4	X	X	X		
Figure 10 State 1	X	X			
State 2		X	X		
State 3	X	X			
Figure 13 State 1	X	X			
State 2		X			
State 3		X	X		
State 4		X	X		
Figure 18 State 1	X	X			
State 2				X	
State 3		X	X		

Table 1: Qualification Requirements

States	Demonstration	Test	Analysis	Inspection	Special Qualification
State 4		X	X		
Figure 23 State 1		X			
State 2		X			
State 3		X			
State 4	X	X			

5.0 REQUIREMENTS TRACEABILITY.

The requirments traceability is provided by the correlation of the functional requirement to the States of the Object classes where that requirement is satisfied. The attendant processing specifications will provide the detail on the computations and data definitions required to satify the requirement.

5.1 GENERIC SHIP HULL

FR1: The ship shall possess a unique location.
The location will be the ship's latitude and longitude, both measured to the nearest second of arc; i.e. degrees, minutes, and seconds.

FR3: The ship shall be capable of steady state movement as measured by the ship's heading and speed.
The ship's speed will be measured to the nearest knot and maintained between maximum and minimum limits per references i. to j.

FR4: The ship shall be capable of maneuvering by changing its heading and speed as ordered by the operator through heading changes, speed changes, and rudder angle changes.
Heading changes will be according to advance and transfer, and turning characteristics per references b and i. through j.
Speed changes will be according to acceleration and deceleration characteristics, including maximum and minimum limits, per references i. through j..

5.2 AIRCRAFT (BOTH F/A-18C AND RED)

FR1: The aircraft shall possess a unique location.
The location will be the aircraft's latitude and longitude, both measured to the nearest second of arc; i.e. degrees, minutes, seconds.

FR3: The aircraft shall be capable of steady state movement as measured by the aircraft's heading , speed, and altitude.
The aircraft's speed will be measured to the nearest knot. The aircraft's speed will be maintained within the maximum and minimum limits per references i. through j.
The aircraft's altitude will be measured to the nearest foot. The aircraft's altitude will be maintained within the maximum and minimum limits per references i. through j.

FR4: The aircraft shall be capable of maneuvering by changing its heading , speed and altitude as ordered by the operator through heading changes, speed changes, and altitude changes.
Heading changes will be according to references b and i. through j.
Speed changes will be according to maximum and minimum limitations per references h. through j.
Altitude changes will be according to the employed climb and dive characteristics, including maximum and minimum limitations, per references i. through j.

5.3 IFF (IDENTIFICATION FRIEND OR FOE) SYSTEM

5.3.1 IFF INTERROGATOR/RECEIVER

Satisfaction: Figure 7 State 2

FR1: The operator will possess the capability to turn the interrogator “on” or “off”.
If “off” the interrogator will not interrogate .
If “on” the interrogator will interrogate in all modes of interrogation.
Note: The Interrogator/Receiver will be referred to simply as the Interrogator. Verification will be achieved by comparing the individual entity information with the response information.

FR2: The Interrogator’s received information will be by mode and code.

5.3.2 IFF TRANSPONDER

FR1: The operator will possess the capability to turn the transponder “on” or “off”.
If “off” the transponder will not respond to interrogation .
If “on” the transponder will respond to interrogation , replying in all modes set “on” for reply.

FR3: The transponder will reply by individual mode and code, for each mode set to “on” for response.

FR4: Interrogation and response will consider a maximum range limitation, i.e. transponder equipped entities beyond maximum range will not respond to interrogations.
Note: For Build NP the maximum interrogation range and maximum transponder range will be the same; 150 nm.

FR5: Transponder response will consider the line-of-sight horizon limitation, i.e. transponder equipped entities below the line-of-sight horizon cut-off will not respond to interrogations.

FR6: When the IFF Transponder is “on”, the operator will have the capability to turn individual modes “on” and “off”.
Modes turned “off” will not respond to interrogations.
Modes turned “on” will respond to interrogations. The response will include the mode and code of the reply. Since the interrogator always interrogates all modes, all modes shall be capable of replying to the same interrogation; any combination of modes 1, 2, 3/A, C, and 4 shall be permitted.
Note: For Build NP Mode 4 will always be on.

Table 2: Traceability Requirements

Functional Requirements	States
Ship	
FR 1	Figure 5, State 1
FR 3	Figure 5, State 2
FR 4	Figure 5, State 3,4
Aircraft	
FR 1	Figure 10, State 1
FR 3	Figure 10, State 2
FR 4	Figure 10, State 2, 3
IFF Interrogator	
FR 1	Figure 13, State 1,2
FR 2	Figure 13, State 3,4
IFF Transponder	
FR 1	Figure 18, States 1, 2
FR 3	Figure 18, State 3, 4 Figure 23, State 1,2, 3, 4
FR 4	Figure 18, State 3
FR 5	Figure 21, State 3
FR 6	Figure 18, State 1

6.0 NOTES.

Acronyms

ALSP	Aggregate Level Simulation Protocol
ASTAB	Automatic Status Board
ADFD	Action Data Flow Diagram
C4I	Command, Control, Communications, Computers and Intelligence
CECOM	US Army Communications-Electronics Command
COTS	Commercial-Off-The-Shelf
CSCI	Computer Software Configuration Item
DBDD	Database Design Description
DCI	DIS/ALSP Communications Interface
DIS	Distributed Interactive Simulation
DSI	Defense Simulation Internet
FMS	Force Modeling and Simulation/C4I
GOTS	Government-Off-The-Shelf
GUI	Graphical User Interface
HPC	High-Performance Computing
HSI	Human-System Integration
HCI	Human Computer Interface
HCI-U	HCI CSCI to User
HWCI	Hardware Configuration Item
IDD	Interface Design Description
ISO	International Standards Organization
JMSS	JSIMS Maritime Software Segment
JPO	Joint Programs Office
JSIMS	Joint Simulation System
LAN	Local Area Network

M&S	Modeling and Simulation
MVC	Model-View-Controller
NCCOSC	Naval Command, Control and Ocean Surveillance Center
NRaD	NCCOSC RDTE DIV
NTDS	Naval Tactical Data System
OO	Object-Oriented
OOA	Object-Oriented Analysis
OOD	Object-Oriented Design
OOT	Object-Oriented Technology
OSI	Open Systems Interconnection
PDES	Parallel Discrete Event Simulation
PDU	Protocol Data Unit
PIP	Project Implementation Plan
SDD	Software Design Description
SM	Simulation Model
SM-DCI	SM CSCI to DCI CSCI
SM-HCI	SM CSCI to HCI CSCI
SPEEDES	Synchronous Parallel Emulation Environment for Discrete Event Simulation
SPP	Scaleable Parallel Processor
SSDD	System/Subsystem Design Description
WAN	Wide Area Network

7.0 APPENDIXES.

Appendix A

```
 ___________________________________
|                                   |
|  Object and Attribute Descriptions |
|___________________________________|
```

```
      Paradigm: shlaer_m
       Project: JMSS NP
       Date: Wed Apr 23 09:16:13 1997


   Output File: obj_desc.txt

=============================================================================
1.  Aircraft (AC)

   Aircraft (AC ID, Type, Current Vector, carries, Is Detected By, Ordered Vector, Climb
Vector, Descent Vector, Accel Vector, Position Time, Max Speed, Min Speed)

   Identifiers:

   Description:
      This is the class for aircraft.

1.1. Aircraft.AC ID

   Description:
      The AC tail number is used for the ID.

   Data_Type:      AC_ID

1.2. Aircraft.Type

   Description:
      The AC sub-type (Red or FA-18).

   Data_Type:      AC_type

1.3. Aircraft.Current Vector

   Description:
      The AC location (Lat, Long), altitude, heading and speed.

   Data_Type:      vector

1.4. Aircraft.carries

   Description:


   Data_Type:      Boolean

1.5. Aircraft.Is Detected By

   Description:
```

Data_Type: Boolean

1.6. Aircraft.Ordered Vector

Description:
The AC's desired (ordered) heading, speed, or altitude.

Data_Type: vector

1.7. Aircraft.Climb Vector

Description:
A vector describing the speed and rate of a climb.

Data_Type: Climb Vector

1.8. Aircraft.Descent Vector

Description:
A vector describing the speed and rate of a descent. (All values are positive).

Data_Type: Climb Vector

1.9. Aircraft.Accel Vector

Description:
The Acel_a and Acel_b parameters. Negative numbers are decelerations.

Data_Type: Accel Vector

1.10. Aircraft.Position Time

Description:
The time (i.e. 1520) at which the position is correct.

Data_Type: Time

1.11. Aircraft.Max Speed

Description:
The AC's maximum speed.

Data_Type: int

1.12. Aircraft.Min Speed

Description:
The AC's minimum speed.

Data_Type: int

===

2. Aircraft Data ()

Aircraft Data ()

Identifiers:

```
   Description:

===============================================================================
3.  Aircraft ID List ()

   Aircraft ID List ()

   Identifiers:

   Description:

===============================================================================
4.  Blue Transponder ()

   Blue Transponder ()

   Identifiers:

   Description:
      The IFF Transponder on an AC.

===============================================================================
5.  Current Time ()

   Current Time ()

   Identifiers:

   Description:

===============================================================================
6.  End of Simulation ()

   End of Simulation ()

   Identifiers:

   Description:

===============================================================================
7.  FA-18 (AC-US)

   FA-18 ()

   Identifiers:

   Description:
      The friendly AC.

===============================================================================
8.  IFF Interrogator (IFF-I)

   IFF Interrogator (ID, OnOff State, is onboard, interrogates, updates)
```

Identifiers:

Description:
This is the IFF Interrogator object.

8.1. IFF Interrogator.ID

Description:
The IFF Interrogator uses the platform object ID as its own ID.

Data_Type:

8.2. IFF Interrogator.OnOff State

Description:
This is the power on/off state for the IFF Interrogator.

Data_Type: Boolean

8.3. IFF Interrogator.is onboard

Description:

Data_Type:

8.4. IFF Interrogator.interrogates

Description:

Data_Type:

8.5. IFF Interrogator.updates

Description:

Data_Type:

==

9. IFF Interrogator Data ()

IFF Interrogator Data ()

Identifiers:

Description:

==

10. IFF Transponder (IFF-T)

IFF Transponder (ID, responds to, is onboard, Type, OnOff State, Reply List, Modes)

Identifiers:

Description:
The Transponder part of the IFF.

10.1. IFF Transponder.ID

Description:
The Transponder box ID number.

Data_Type: IFF_ID

10.2. IFF Transponder.responds to

Description:

Data_Type: Boolean

10.3. IFF Transponder.is onboard

Description:

Data_Type: Boolean

10.4. IFF Transponder.Type

Description:
The type of IFF Transponder (Blue, Red, Ship).

Data_Type: IFF Transponder

10.5. IFF Transponder.OnOff State

Description:
Tells whether this Transponder is on or off.

Data_Type: Boolean

10.6. IFF Transponder.Reply List

Description:
Provides the data structure of the Transponder reply.

Data_Type: IFF Reply

10.7. IFF Transponder.Modes

Description:
This data structure tells which modes are on.

Data_Type: IFF Modes

===

11. IFF Transponder Data ()

IFF Transponder Data ()

Identifiers:

Description:

==
12. IFF-I Data ()

IFF-I Data ()

Identifiers:

Description:

==
13. Red A/C (AC-R)

Red A/C ()

Identifiers:

Description:
The enemy AC.

==
14. Red Transponder ()

Red Transponder ()

Identifiers:

Description:
The IFF Transponder on a Red AC.

==
15. Set AC Timer ()

Set AC Timer ()

Identifiers:

Description:

==
16. Set Done Timer ()

Set Done Timer ()

Identifiers:

Description:

==
17. Set IFFI Timer ()

Set IFFI Timer ()

Identifiers:

Description:

==
18. Ship (S)

Ship (Ship ID, Current Vector, Detects, Is Detected By, Contains, Creates, Ordered Vector, Position Time, Max Speed, Min Speed, Acceleration, Slow Maneuver Speed, Max Rudder)

Identifiers:

Description:
This is the class for ships.

18.1. Ship.Ship ID

Description:
The USS Naval number (hull number).

Data_Type: shipID

18.2. Ship.Current Vector

Description:
The current location (Lat, Long), heading (degrees), rudder angle, and speed of the ship.

Data_Type: vector

18.3. Ship.Detects

Description:

Data_Type: Boolean

18.4. Ship.Is Detected By

Description:

Data_Type: Boolean

18.5. Ship.Contains

Description:

Data_Type: Boolean

18.6. Ship.Creates

Description:

Data_Type: Boolean

18.7. Ship.Ordered Vector

Description:

The ship's desired heading and speed.

Data_Type: vector

18.8. Ship.Position Time

Description:
The time (military i.e. 1635) that the current position is valid.

Data_Type: Time

18.9. Ship.Max Speed

Description:
Maximum ship speed (knots).

Data_Type: int

18.10. Ship.Min Speed

Description:
The ship minimum speed (knots).

Data_Type: int

18.11. Ship.Acceleration

Description:
The ship's current acceleration (deceleration is by a minus sign).

Data_Type: real

18.12. Ship.Slow Maneuver Speed

Description:
The ship's speed (kknots) at which it performs "Slow Maneuvering".

Data_Type: int

18.13. Ship.Max Rudder

Description:
The ship;s maximum allowed rudder angle.

Data_Type: int

==

19. Ship Data ()

Ship Data ()

Identifiers:

Description:

==

20. Ship ID List ()

Ship ID List ()

Identifiers:

Description:

==
21. Ship Store ()

Ship Store ()

Identifiers:

Description:

==
22. Ship Transponder ()

Ship Transponder ()

Identifiers:

Description:
The IFF Transponder on a ship.

==
23. Track (T)

Track (Track ID, Ship ID, Target Vector, IFF Mode, Track Status, Target Type, Target ID, Is Updated By, Is Created By)

Identifiers:

Description:
The tracks of objects (both AC and ships) maintained by each ship.

23.1. Track.Track ID

Description:

Data_Type: int

23.2. Track.Ship ID

Description:
The USS Naval ship number.

Data_Type: shipID

23.3. Track.Target Vector

Description:
A data construct that includes the location (including altitude), the heading and the speed.

Data_Type: vector

23.4. Track.IFF Mode

Description:
The current IFF mode setting, or OFF if the IFF is not in one of the operational modes.

Data_Type: iff_mode

23.5. Track.Track Status

Description:
The current status of the track is an enumerated item that describes if the target is identified, friendly, AC ID, or unknown..

Data_Type: ID_status

23.6. Track.Target Type

Description:
The type (AC, ship) of the target. The type may include the sub-type (Red AC, FA-18) if known.

Data_Type: target_type

23.7. Track.Target ID

Description:
The ID of the target (if provided by the IFF), such as the AC tail number or the ship USS Naval number.

Data_Type: targetID

23.8. Track.Is Updated By

Description:

Data_Type: Boolean

23.9. Track.Is Created By

Description:

Data_Type: Boolean

==

24. Track Data ()

Track Data ()

Identifiers:

Description:

Appendix B

Object Event List

```
       Paradigm: shlaer_m
        Project: JMSS NP
        Date: Wed Apr 23 09:14:06 1997


   Output File: OEL_daves-np.txt


Event Name              |Event Data              |Source                  |Destination
========================|========================|========================|======================
==|
AC3                     |                        |Aircraft
|asd                     |
                        |                        |Aircraft                |AC Vector
Reporting     |
                        |                        |Aircraft                |Aircraft
Flying         |
                        |                        |Aircraft                |Initializing
Aircraft   |
                        |                        |Aircraft                |End of
Simulation       |
------------------------|------------------------|------------------------|----------------------
--|
Sim2                    |                        |Aircraft
|asd                     |
                        |                        |Aircraft                |AC Vector
Reporting     |
                        |                        |Aircraft                |Aircraft
Flying         |
                        |                        |Aircraft                |Initializing
Aircraft   |
                        |                        |Aircraft                |End of
Simulation       |
------------------------|------------------------|------------------------|----------------------
--|
AC9                     |                        |Aircraft
|asd                     |
                        |                        |Aircraft                |AC Vector
Reporting     |
                        |                        |Aircraft                |Aircraft
Flying         |
                        |                        |Aircraft                |Initializing
Aircraft   |
                        |                        |Aircraft                |End of
Simulation       |
------------------------|------------------------|------------------------|----------------------
--|
AC2                     |                        |Aircraft
|asd                     |
                        |                        |Aircraft                |AC Vector
Reporting     |
                        |                        |Aircraft                |Aircraft
Flying         |
                        |                        |Aircraft                |Initializing
```

```
Aircraft   |
                        |                        |Aircraft                |End of
Simulation       |
------------------------|------------------------|------------------------|----------------------
--|
IFF0                    |                        |IFF Interrogator        |Interrogator
Off        |
                        |                        |IFF Interrogator
|Idle                    |
                        |                        |IFF Interrogator        |Send
Challenges         |
                        |                        |IFF Interrogator        |Evaluate
Responses      |
------------------------|------------------------|------------------------|----------------------
--|
Done                    |                        |IFF Interrogator        |Interrogator
Off        |
                        |                        |IFF Interrogator
|Idle                    |
                        |                        |IFF Interrogator        |Send
Challenges         |
                        |                        |IFF Interrogator        |Evaluate
Responses      |
                        |                        |IFF Transponder         |Transponder
Off         |
                        |                        |IFF Transponder         |Respond to IFF
Interroga|
                        |                        |IFF Transponder
|Idle                    |
                        |                        |IFF Transponder         |Set
Mode                |
------------------------|------------------------|------------------------|----------------------
--|
IFF5                    |                        |IFF Transponder         |Transponder
Off         |
                        |                        |IFF Transponder         |Respond to IFF
Interroga|
                        |                        |IFF Transponder
|Idle                    |
                        |                        |IFF Transponder         |Set
Mode                |
------------------------|------------------------|------------------------|----------------------
--|
IFF2                    |                        |IFF Transponder         |Transponder
Off         |
                        |                        |IFF Transponder         |Respond to IFF
Interroga|
                        |                        |IFF Transponder
|Idle                    |
                        |                        |IFF Transponder         |Set
Mode                |
------------------------|------------------------|------------------------|----------------------
--|
IFF6                    |                        |IFF Transponder         |Transponder
Off         |
                        |                        |IFF Transponder         |Respond to IFF
Interroga|
                        |                        |IFF Transponder
|Idle                    |
                        |                        |IFF Transponder         |Set
Mode                |
------------------------|------------------------|------------------------|----------------------
--|
```

Appendix C

```
 ______________________________________
|                                      |
 |         Relationship Report        |
|______________________________________|

      Paradigm: shlaer_m
       Project: JMSS NP
          Date: Wed Apr 23 08:54:39 1997


   Output File: RD_daves-np.txt

1.
==========================================================================
R5 is a relation between the following classes:
   Class1: Aircraft
      with Role Name: is onboard
   Class2: IFF Transponder
      with Role Name: carries
   Has the following Cardinality: EXACTLY ONE:EXACTLY ONE

   Description:      The relationship between an AC and its onboard IFF
Transponder. Each AC has onboard one and only one transponder.

--------------------------------------------------------------------------

2.
==========================================================================
 : Aircraft is a superclass of: FA-18
--------------------------------------------------------------------------

3.
==========================================================================
AC Type : Aircraft is a superclass of: Red A/C
--------------------------------------------------------------------------

4.
==========================================================================
R4 is a relation between the following classes:
   Class1: IFF Interrogator
      with Role Name: responds to
   Class2: IFF Transponder
      with Role Name: interrogates
   Has the following Cardinality: MANY:MANY CONDITIONAL

   Description:

--------------------------------------------------------------------------

5.
==========================================================================
```

```
 : IFF Transponder is a superclass of: Blue Transponder
--------------------------------------------------------------------------

6.
==========================================================================
Transponder Type : IFF Transponder is a superclass of: Red Transponder
--------------------------------------------------------------------------

7.
==========================================================================
 : IFF Transponder is a superclass of: Ship Transponder
--------------------------------------------------------------------------

8.
==========================================================================
R3 is a relation between the following classes:
   Class1: Ship
      with Role Name: is onboard
   Class2: IFF Transponder
      with Role Name: Contains
   Has the following Cardinality: EXACTLY ONE:EXACTLY ONE

   Description:      The relationship of a ship to its onboard IFF
Transponder. Each ship has one and only one IFF Transponder.

--------------------------------------------------------------------------

9.
==========================================================================
 is a relation between the following classes:
   Class1: Ship
      with Role Name: undefined
   Class2: Ship
      with Role Name: undefined
   Has the following Cardinality: EXACTLY ONE:EXACTLY ONE

   Description:

--------------------------------------------------------------------------

10.
==========================================================================
R7 is a relation between the following classes:
   Class1: Ship
      with Role Name: Is Created By
   Class2: Track
      with Role Name: Creates
   Has the following Cardinality: EXACTLY ONE:MANY CONDITIONAL

   Description:      This is the relationship between a Ship and Track(s).

--------------------------------------------------------------------------

11.
```

```
==========================================================================
R2 is a relation between the following classes:
   Class1: Ship
      with Role Name: is onboard
   Class2: IFF Interrogator
      with Role Name: Contains
   Has the following Cardinality: EXACTLY ONE:EXACTLY ONE

   Description:

--------------------------------------------------------------------------

12.
==========================================================================
identifies A/C by is a relation between the following classes:
   Class1: Track
      with Role Name: undefined
   Class2: Ship
      with Role Name: undefined
   Has the following Cardinality: EXACTLY ONE:EXACTLY ONE

   Description:

--------------------------------------------------------------------------

13.
==========================================================================
is identified by is a relation between the following classes:
   Class1: Track
      with Role Name: undefined
   Class2: Aircraft
      with Role Name: undefined
   Has the following Cardinality: EXACTLY ONE:EXACTLY ONE

   Description:

--------------------------------------------------------------------------
```